AD-A057 130 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS ANALYZER (URA) USER'S MANUAL H6180/MULTICS/VE--ETC(U)
MAR 77 D TEICHROEW, A HERSHEY, S SPEWAK F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-128 NL

1 OF 6
AD A057130



AD-A057130

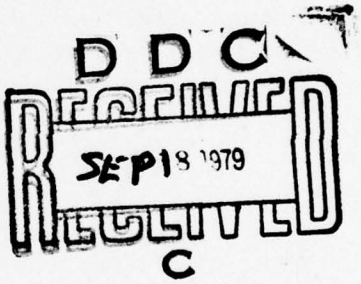
ESD-TR-78-128



USER REQUIREMENTS ANALYZER (URA)
USER'S MANUAL
H6180/MULTICS/VERSION 3.2 & FIGURES

D. Teichroew
A. Hershey
S. Spewak
University of Michigan
Department of Industrial & Operations Engineering
Ann Arbor, MI 48109

March 1977



Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

79 09 17 047

LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

This Technical Report has been reviewed and is approved for publication.

CHARLES J. HARTMAN
Project Engineer

CHARLES J. GREWE, JR., Lt Col, USAF
Chief, Technology Applications Division

JOHN T. HOLLAND, Lt Col, USAF
Chief, Standards & Evaluation Division

FOR THE COMMANDER

STANLEY P. DERESKA, Colonel, USAF
Director, Computer Systems Engineering
Deputy for Technical Operations

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-78-128	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) User Requirements Analyzer (URA) User's Manual H6180/Multics/Version 3.2 & Figs		5. TYPE OF REPORT & PERIOD COVERED July 1976 to March 1977 Manual
		6. PERFORMING ORG. REPORT NUMBER CDRL Item 003
7. AUTHOR(s) D. Teichroew A. Hershey S. Spewak		8. CONTRACT OR GRANT NUMBER(s) F19628-76-C-0197
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Michigan Department of Industrial & Operations Engineering Ann Arbor, Michigan 48109		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE 64740F Project 2237
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Technical Operations Electronic Systems Division Hanscom AFB, MA 01731		12. REPORT DATE March 1977
		13. NUMBER OF PAGES 358
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer-Aided Design
Information Processing
Information System Requirements
Requirements Analysis
Requirements Language
Requirements Specification
Specification Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is part of a series that deals with a Computer-Aided Design and Specification Analysis Tool (CADSAT). The purpose of the tool is to describe the requirements for information processing systems and to record such descriptions in machine processable form. The major components of CADSAT are the User Requirements Language (URL) and the User Requirements Analyzer (URA) which can operate in an interactive computer environment. This report describes the capabilities and the commands associated with URA and its operation on a Honeywell 6180 Multics Computer.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Table of Contents

PART I USER REQUIREMENTS ANALYZER

	PAGE
1. Introduction	4
2. Using URA	5
2.1 The URA Command Language	5
2.2 Command Parameters	5
2.3 Sequence of Commands	6
2.4 The HELP Command	6
3. The URA Environment	7
3.1 Initiating URA	7
3.2 Batch versus On-Line Use of URA	8
4. Specifying Input to URA Commands	11
4.1 The NAME Parameter	11
4.2 The FILE and INPUT Parameters	11
4.3 Entering Data into an Input File	12
4.4 Using NAME-GEN	12
4.5 Using PUNCH Files	12
5. Receiving Output from URA Commands	13
5.1 The NOPRINT Parameter	14
5.2 The INDEX Parameter	14
5.3 The NOSOURCE and XREF Parameters	15
6. Control Commands	15
7. Modifier Commands	15
7.1 CHANGE-TYPE	19
7.2 DELETE	21
7.3 DELETE-COMMENT-ENTRY	23
7.4 DELETE-PSL	25
7.5 INPUT-PSL	27
7.6 PUNCH-COMMENT-ENTRY	30
7.7 RENAME	32
7.8 REPLACE-COMMENT-ENTRY	34
8. Report Commands	35
9. Error Diagnostics	36
10. How to Correct Errors	70
10.1 Input Errors	70
10.2 Logical Errors	76

Table of Contents

PART II USAGE OF THE USER REQUIREMENT ANALYZER UNDER MULTICS

	PAGE
1. Introduction	80
2. Using Multics	80
3. The URA Environment	80
4. Specifying Input to URA Commands	81
4.1 Entering Data into a Data Set	81
4.2 Specifying Input Data Interactively	81
5. Receiving Output from URA Commands	82
5.1 Using the Output Parameter	82
5.2 Using Punch Segments	82
6. Control Commands	82
6.1 Set Command	82
6.2 Display Command	82
6.3 Stop Command	83
7. Modifier Commands	83
8. Report Commands	83
9. Error Conditions	84
9.1 Initial Messages	84
9.2 Abnormal Termination	84
9.3 Data Base Already Open	84

Table of Contents

PART III URA OUTPUTS

	PAGE
1. Introduction	86
2. Objects of Reports from a URA Data Base	86
2.1 Purpose of URA Reports	86
2.2 Relation of URA Reports to Logical System Design	87
2.3 Advantages of Using URA Reports	88
3. Generation of Reports	91
3.1 URA Command Language for Reports	91
3.2 Retrieval of Information from the URA Data Base	92
ATTRIBUTE REPORT	95
CONSISTS COMPARISON REPORT	97
CONSISTS MATRIX REPORT	100
CONTENTS REPORT	104
DATA PROCESS REPORT	108
DICTIONARY REPORT	118
EXTENDED PICTURE REPORT	121
FORMATTED PROBLEM STATEMENT	128
FREQUENCY REPORT	153
IDENTIFIER INFORMATION REPORT	155
KWIC INDEX	159
NAME GEN	161
NAME LIST	168
PICTURE	170
PROCESS CHAIN REPORT	185
PROCESS INPUT/OUTPUT	190
PUNCHED COMMENT ENTRIES	194
RESOURCE CONSUMPTION ANALYSIS REPORT	196
SECURITY ANALYSIS REPORT	200
STRUCTURE	202
SUMMARY OF THE DATA BASE	204
INDEX	206

Table of Contents

PART IV USER REQUIREMENTS ANALYZER COMMAND DESCRIPTIONS

	PAGE
A Note on Version 3.2 of URA	209
The URA Command Language	210
URA Command Language Installation Dependencies	211
Command Language Syntax Notation	211
Format of Command Descriptions	214
CHANGE-TYPE	217
CONSISTS-COMPARISON	219
CONSISTS-MATRIX	220
CONTENTS	221
DATA-PROCESS	223
DELETE	225
DELETE-COMMENT-ENTRY	226
DELETE-PSL	228
DICTIONARY	229
ENTITY-IDENTIFIER	231
EXTENDED PICTURE	232
FORMATTED-PROBLEM-STATEMENT	235
FREQUENCY	239
HELP	240
INPUT-PSL	241
KWIC	242
NAME-GEN	243
NAME-LIST	251
PICTURE	252
PRINT-ATTRIBUTE-VALUES	254
PROCESS-CHAIN	255
PROCESS-INPUT-OUTPUT	257
PUNCH-COMMENT-ENTRY	259
RENAME	261
REPLACE-COMMENT-ENTRY	262
RESOURCE CONSUMPTION ANALYSIS	263
SECURITY ANALYSIS	265
STRUCTURE	266
SUMMARY	267

Table of Contents

PART V AUTOMATED DOCUMENTATION SYSTEM USERS' MANUAL

	PAGE
INTRODUCTION	269
1.0 DOCUMENT INITIALIZATION	270
1.1 Strategy for the Document Initialization and General Process	270
1.2 Syntax for Entries in the Documentation Schema	273
2.0 DOCUMENTATION DATA BASE POPULATION	276
2.1 Strategy for the Documentation Data Base Population	276
2.2 Syntax for the Documentation Data Base	277
3.0 THE DOCUMENTATION GENERATION	280
4.0 USAGE OF THE DOCUMENTATION GENERATOR	281
4.1 Development of the Documentation Schema	281
4.2 Development of the Documentation Source	286
4.3 Development of the Analyzer Data Base With the Document Generator in Mind	292

Table of Contents

	PAGE
Appendix A: URA Command Abbreviations	295
Appendix B: Creating and Initializing URA Data Bases	303
Appendix C: URA Utilities	305
Appendix D: Reporting Problems With URA	308
Appendix E: Using the URA Command Language with Multics	310
Appendix F: Example of Document Schema and Source	327
Appendix G: Document Examples	339
Appendix H: Executing Automatic Documentation Systes	342
Appendix I: URA Character Set	347

Figures

		PAGE
1.	Interaction Between Operating System and URA Processing Modes	4a
2.	CHANGE-TYPE Report	20a
3.	CHANGE-TYPE Report	20b
4.	CHANGE-TYPE Report	20c
5.	CHANGE-TYPE Report	20d
6.	DELETION Report	22a
7.	DELETION Report	22b
8.	DELETED COMMENT ENTRIES Report	24a
9.	DELETED COMMENT ENTRIES Report	24b
10.	DELETED URL Report	26a
11.	AS-IS SOURCE LISTING	28a
12.	AS-IS SOURCE LISTING AND CROSS REFERENCE	28b
13.	PUNCHED COMMENT ENTRIES Report	31a
14.	PUNCHED COMMENT ENTRIES Report	31b
15.	PUNCHED COMMENT ENTRIES Report	31c
16.	RENAME Report	33a
17.	RENAME Report	33b
18.	REPLACED COMMENT ENTRIES	35a
19.	ATTRIBUTE Report	96a
20.	CONSISTS COMPARISON Report	99a
21.	CONSISTS MATRIX Report	103a
22.	CONSISTS MATRIX report	103e
23.	CONTENTS Report	107a
24.	CONTENTS Report	107b
25.	DATA PROCESS Report	117a
26.	DATA PROCESS Report	117f
27.	DICTIONARY Report	120a
28.	DICTIONARY Report	120b
29.	EXTENDED PICTURE Report	126a
30.	EXTENDED PICTURE Report	126e
31.	FORMATTED PROBLEM STATEMENT Margin Parameters	129
32.	FORMATTED PROBLEM STATEMENT	152a
33.	FORMATTED PROBLEM STATEMENT	152b
34.	FREQUENCY Report	154a
35.	IDENTIFIER INFORMATION Report	158a
36.	KWIC INDEX Report	160a
37.	NAME GEN Report	167a
38.	NAME GEN Report	167b
39.	NAME GEN Report	167c
40.	NAME GEN Report	167d
41.	NAME LIST	169a
42.	General PICTURE Format and Limits per Page	171
43.	INTERFACE PICTURE Report	173
44.	SET PICTURE Report	174
45.	INPUT PICTURE Report	175
46.	OUTPUT PICTURE Report	176
47.	ENTITY PICTURE Report	177
48.	GROUP/ELEMENT PICTURE Report	178
49.	PROCESS PICTURE Report	179
50.	PICTURE Report	181a
51.	PICTURE Report	181d
52.	PICTURE Report	181f
53.	PROCESS CHAIN Report	188a
54.	PROCESS INPUT/OUTPUT Report	192a
55.	PROCESS INPUT/OUTPUT Report	192b
56.	PUNCHED COMMENT ENTRIES	195a

Figures

		PAGE
57.	RESOURCE CONSUMPTION ANALYSIS Report	199a
58.	RESOURCE CONSUMPTION ANALYSIS Report	199c
59.	SECURITY ANALYSIS Report.	201a
60.	INPUT STRUCTURE Report	203a
61.	OUTPUT STRUCTURE Report	203b
62.	INTERFACE STRUCTURE Report	203c
63.	PROCESS STRUCTURE Report	203d
64.	PROCESSOR Report	203f
65.	DATA BASE SUMMARY Report	205a
66.	PROCESS INPUT/OUTPUT	207a
67.	Documentation Initializer	272
68.	Documenation Data Base Populator	279
69.	Documentation Generator	280a

Tables

PAGE

1. Comparison of Outline versus Batch Use of URA 9

2. Types of Information Taken and Presented by URA Reports ... 94

3. Structure Relationships Displayed in Extended Picture Report 122

4. Data Flow Relationships Displayed in Extended Picture Report 123

5. Completeness Checks that may be made by Visual Analysis of the EXTENDED PICTURE Report 127

6. Ordering for FORMATTED PROBLEM STATEMENT 130

7. Name Types and Relationships Presented in PICTURE Report .. 182

8. Completeness Checks that may be made by Using the PICTURE Report 184

9. Completeness Checks that may be made by Visual Analysis of the Process Chain Report 189

B-1 Multics Default Segment Names 325

PART I USER REQUIREMENTS ANALYZER

The goal of Part I is to assist the User Requirements Analyzer (URA) user in effectively using the URA Modifier Commands specified in Part IV, "User Requirements Analyzer Command Descriptions." This paper illustrates the usage of Version 3.2 of the User Requirements Analyzer and specifies the steps in creating the URA data base, inputting User Requirements Language (URL) statements, modifying the contents of the data base, generating URA outputs, and correcting syntactical and logical errors.

Since URA is used in conjunction with an operating system, this part should be used in conjunction with Part II which presents installation dependent features to be considered when using URA. This part covers installation independent features of URA, general concepts, etc. Each section of this part has a corresponding section in Part II. This allows easy reference between general concepts and the actual practice of applying them in a particular installation.

PART II USAGE OF THE USER REQUIREMENTS ANALYZER UNDER MULTICS

The purpose of Part II is to assist the URA user in effectively manipulating the URA command language under Multics. This paper covers those installation dependent features of URA and is intended to be used in conjunction with the User Requirements Analyzer, Part I. Each section of this Part has a corresponding section in the User Requirements Analyzer. This Part illustrates the usage of Version 3.2 of the User Requirements Analyzer.

PART III URA OUTPUTS

The goals of Part III are to assist the User Requirements Analyzer user in generating reports from the information in a URA data base, describe the standard reports available in URA, and finally, provide general guidelines on using these reports to aid in the logical system design process. In order to generate the reports described in this paper it is necessary to understand the information presented in Part I and Part II for the installation in which URA is being used. It is also desirable to use the "User Requirements Analyzer Commands Descriptions," Part IV as a reference for a better understanding of the URA commands and parameters used to generate a particular report.

This part describes those URA reports available in Version 3.2 of the User Requirements Analyzer.

PART IV USER REQUIREMENTS ANALYZER COMMAND DESCRIPTIONS

The objective of Part IV is to give the user of the User Requirements Analyzer (URA) the list of the commands available, the correct syntax of these commands, and the parameters allowable for each command. This part is not intended as a handbook on how to use URA, but rather as a reference for the commands available. Part I and Part II describe how to use these commands and present a detailed description of each of the outputs generated by these commands. This paper describes the facilities of Version 3.2 of the User Requirements Analyzer.

The manner in which URA directly interfaces with a particular operating system is given in Appendix E. Also included in this Appendix is a formal description of the URA commands available only under that particular operating system.

PART V AUTOMATED DOCUMENTATION SYSTEM USER'S MANUAL

The objective of Part V is to provide detailed operating instructions for the Automated Documentation System (ADS). ADS is a stand-alone program and file structure that operates on a URL data base to produce requirements specifications in standard formats (e.g., MIL STD 483).

PART I

USER REQUIREMENTS ANALYZER

1. INTRODUCTION

The first step in use of the URL/URA system consists of specifying a problem statement in URL statements. The second step consists of using URA to enter the problem statement into a computer data base. URA extracts information from the URL statements and stores it in a URA data base. Once this information (a problem statement) is in the data base, it can be modified, new information can be added to it, and reports presenting the status of the problem statement can be generated. These actions are implemented by the URA commands available in the URA processing mode. This mode of operation may be attained by accessing the URA software available on a particular operating system. Therefore, by understanding the operating commands that interact with URA and the URA command language, the problem definer (user) can effectively manipulate the contents of a URA data base. Part I, of this manual, serves as a guide to using the URA commands.

Operating system and URA commands can only be used in their respective processing modes. Operating System commands can be used from time of signing on to the system, to the time access to the URA software is acquired. At this point, only URA commands may be used until the problem definer returns control to the operating system or terminates processing to be done in URA mode (through use of the URA "STOP" command). Operating system commands then can again be used up to the time of signing off. This interaction between operating system and URA modes is illustrated by Figure 1. An exception to this rule is the URA command "MTS" which allows one to execute an operating system command while in URA mode.

The first five sections of Part I deal with URA at an introductory level. Section 2 presents introductory information about the use of URA. Section 3 explains the procedure of initializing URA once on the operating system. Sections 4 and 5 present practical concepts and conventions to be known before using URA. (Once access to URA has been achieved, various commands are available; these commands are described in Section 6, 7 and 8.) Several examples are given in these sections in order to better illustrate the results of specific implementations. Sections 9 and 10 deal with handling errors encountered in the use of URA. Appendix A presents a list of all URA commands available (and the parameters for each command) as well as the abbreviations for all these to serve as a quick reference. Throughout Part I the long forms of URA commands and parameters are used interchangeably with their abbreviations.

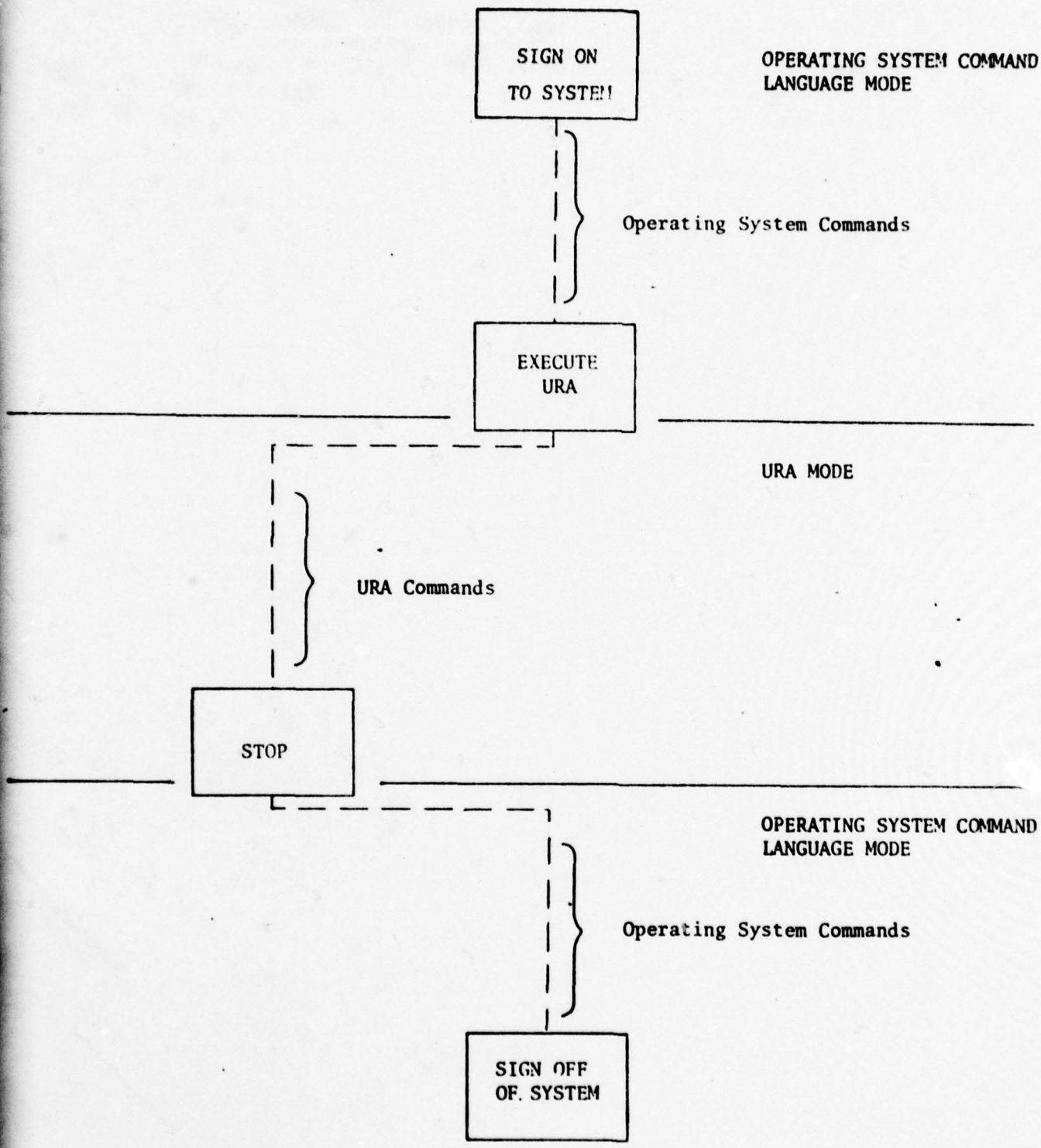


Figure 1: Interaction between Operating System and URA processing modes

2. USING URA

2.1 The URA Command Language

The URA Command Language commands may be grouped into three major types.

Control Commands

Control Commands are used to pass certain control information to the User Requirements Analyzer. These commands are operating system dependent and are given in Part II of this manual. The use of these commands does not change the contents of the data base.

Modifier Commands

Modifier Commands are used to modify the contents of the URA data base in the manner defined by the problem definer. These commands take legal URL statements or URL names as input. URA then generates error diagnostics as well as an output report to present the outcome of the data base alteration.

Report Commands

Report Commands retrieve data from the URA data base and output it in some standard format. These reports do not change the contents of the data base whatsoever. Their purpose is solely that of displaying some or all of the contents of the data base.

2.2 Command Parameters

Most URA commands have parameters which may be set by the user to modify the actions of the command. There are basically five types of parameters:

Input data parameters - these parameters specify the data to be used as input to the command. INPUT, FILE and NAME are examples of Input data parameters.

Input control parameters - these parameters specify how the input data is to be used, changed, etc., by the command. The TYPE parameter for the CHANGE-TYPE command and CONTAINED/CONSISTS parameter for the CONSISTS-MATRIX command are examples of this type.

Output data parameters - these parameters specify if output is to be generated from the command and the form in which it is presented. The PUNCH and PRINT parameters are examples

of this type of parameter.

Output option parameters - these parameters specify options which may be included or omitted from the output. The LEVELS parameter for the CONTENTS command and the DESCRIPTION parameter for the DICTIONARY command are examples of this type.

Output format parameters - these parameters specify alternate formats for presenting the information in the output from the command. The NEW-PAGE parameter and HMARG parameter for the FPS command are examples of this type.

2.3 Sequence of Commands

Although all of the commands can be issued independently of each other, it is often advantageous to use some commands in sequence since the output of one command may be used as input by another. The most common instance of this is when NAME-GEN is used to select certain names (say all PROCESSES for example) which can then be used as input to a Report Command (possibly PICTURE, for a PICTURE REPORT for all PROCESS names). Though NAME-GEN is technically classified as a Report Command one of its major functions is to selectively retrieve names stored in the data base. PUNCH-COMMENT-ENTRY and REPLACE-COMMENT-ENTRY often occur in sequence. For more information about use of these commands, see "URA Outputs"[1] and "URA Command Desriptions."[2]

2.4 The HELP Command

The HELP Command provides the user with information about the syntax and parameters of URA commands. The HELP command does not affect the manner in which URA operates nor accesses information in the data base. For this reason, it is not classified as a Control, Modifier or Report Command. When the HELP command is given, URA displays a list of all available URA commands and their abbreviations. By specifying a particular URA command name as a parameter to the HELP command:

```
HELP CONTENTS
```

for example, all the parameters available for this command will be printed. If the "LONG" parameter were given in conjunction with a command name:

```
HELP FPS LONG
```

---

[1] Part III
[2] Part IV

all parameters for the FORMATTED-PROBLEM-STATEMENT command would be printed out as well as a description of the function of each of these parameters for the command. This description is presented in the same format as that in "User Requirements Analyzer Command Descriptions."[1] To illustrate an example, when "HELP CONTENTS" was given the following information was printed:

```
CONTENTS

Prototype: CONTENTS(CONT) [parameter] ...

Parameters:

     FILE[=fdname], NAME(N)=user name Default: FILE
     INDEX, NOINDEX                   Default: NOINDEX
     LEVELS=integer, LEVELS=ALL       Default: ALL
     NCFLAG, NONCFLAG                 Default: NONCFLAG
```

3. THE URA ENVIRONMENT

The considerations necessary for using URA and preparing for using URA define the URA environment. The following points must be considered:

- How the data base is prepared.
- How URA is executed.
- How URA is used.
- How batch use of URA is different from terminal use.

The first three points are presented in section 3.1, "Initiating URA," and the last point is presented in section 3.2, "Batch Versus on-line Use of URA."[2]

3.1 Initiating URA

The steps required to prepare a URA data base for access by URA are:

i) The URA data base file must be created.

Creation of the data base file occurs only once. Once

---

[1] Part IV

[2] The exact commands to carry out the above procedures are installation-dependent and can be found in Part II (Section 3.1).

created, any changes to be made (including emptying the whole file) can be made without destroying it.

ii) The data base must then be initialized in order to be accessed by URA.

Initialization, for the most part, occurs only once, at the time of creating the data base. The data base must be reinitialized if emptied.

Executing URA involves running the URA program. This allows the user to specify URA commands to change the contents of the data base or generate reports about its contents.

Once the URA program has been invoked the following steps are suggested in using URA:

i) The data base to be accessed by the URA commands must be specified.

This should be done any time URA is used to ensure that the user is accessing the correct data base. In some cases the data base to be used will set by default any time the user executes URA. Even if this is the case, the user should be aware that the default is in effect.

ii) URA commands are given to modify, update and/or generate reports on the data base information.

Any of the commands given in "User Requirements Analyzer Command Descriptions"[1] can be issued to accomplish these tasks. The order in which commands are given and which commands are given is determined by the user.

iii) The STOP command is given to terminate the URA session.

3.2 Batch Versus On-line Use of URA

The manner in which a user interacts with URA via batch processing differs from on-line (terminal) usage. There are various advantages and disadvantages to either approach. A few of these are given in Table I.

The procedures given in section 3.1 are the same for both on-line and batch use of URA. The specific manner in which URA commands are given by the user, however, is different.

In batch processing of URA commands, URA commands are given, one per card, following the card which executes the URA program.

---

[1] Part IV

Any errors detected in specifying the command (or its parameters) cause the command to be ignored, and URA then moves to the next command.

When executing URA in on-line mode, the user must wait for the message:

ENTER COMMAND (AND ANY PARAMETERS)

before giving any commands. After typing in the command followed by a carriage return, the user must wait again until the command prompting message is issued before entering the next command. If an error occurs when specifying the command (or its parameters) the user will be prompted for a replacement.

On-line Use of URA

ADVANTAGES

- The user is able to handle errors as theyoccur. Errors can be corrected before anyattempt is made to modifyor retrieve information in the data base.
- Which URA commands to be issued and the order in which they are given does not have to be predetermined, i.e., the user may issue commands ad hoc.
- Utilizing the edit facility of the operating system allows the user to make changes to information in the data base quickly and efficiently.

DISADVANTAGES

- Loss of connection between terminal and computer (line hits) may cause the contents of the data base to become unusable. Recovery procedures would be required to restore the data base contents.
- On-line use is generally more expensive than batch because of connect time costs and other additional costs related to terminal access.

Batch Use of URA

ADVANTAGES

- Requires the URA user to think out procedures (list of URA commands) to be executed before they are executed.
- Cheaper to run than on-line use.
- All output generated by URA is printed in a usable format,

i.e., via the line printer. Output generated at the terminal may be part of the terminal listing and interspersed with other information.

Table 1. Comparison of On-Line Versus Batch Use of URA

DISADVANTAGES

- Turn-around time for jobs may be long.
- Editing of information in the data base may require two batch runs; one to retrieve, one to replace.
- Errors are ignored and any subsequent URA commands would be run regardless.

Table 1. Continued

4. SPECIFYING INPUT TO URA COMMANDS

For most URA commands, one or more names (specified by the user) can be used as "input" to the command. This can be done by utilizing the "input data" parameters for the command. In the case of Modifier Commands, the modification is made for each name used as input. For Report Commands, information is retrieved for each of the names used as input. Except for the INPUT-PSL command, all names used as input to the Modifier and Report Commands must be names already stored in the problem definer's URA data base.

4.1 The NAME Parameter

There are two methods of specifying names to be input to a command. The simplest way is to use the NAME parameter. When this parameter is used, the modification will be made, or a report will be generated, for only that name specified by the NAME parameter. For example, if NAME=T-CARD were used for the DELETE command, only T-CARD would be deleted from the data base. Likewise, if NAME=T-CARD were used as a parameter for the CONTENTS command, the CONTENTS REPORT would be generated for the name T-CARD, and no others.

4.2 The FILE and INPUT Parameters

The second way to specify names as input to a URA command is to put all the names for which the modification is to be made, or a report generated, into a file and specify that the contents of that file are to be used as input. At most installations this specification can simply be done via the FILE and INPUT parameters, but varies slightly from one installation to the next. (See Part II.) FILE and INPUT are different in the way names can be formatted within the file specified by these parameters. When using the FILE parameter, each name in the specified file must begin in the first column of each line of the file and only one name per line is allowed. The format for

files specified by the INPUT parameter varies according to the URA command using this parameter. For example, if a file is used as input to the INPUT-PSL command, the file must consist of URL statements to be entered into the URA data base. For those Modifier Commands that allow the INPUT parameter, the particular format needed for the input file is specified in the description of each command.

If one particular file name is used throughout a given terminal session to contain various information and name lists to interact with URA commands, the user should remember to empty the file before each time new data is to be written into it. Otherwise, information used as input to a previous command may be leftover when using the file with subsequent commands.

4.3 Entering Data Into an Input File

The manner in which data is entered into a file is installation-dependent. See Part II, Section 4, for the details of this procedure.

4.4 Using NAME-GEN

One alternative to specifying input to Modifier and Report Commands is to let NAME-GEN generate the input file. The various parameters for NAME-GEN allow selection of particular types of names that are desired (e.g., all GROUPS and ENTITIES) and retrieval of these names which are then put into a file by URA. The names are formatted, one name per line starting in the first column of the line. The contents of this file can be used as input to a Report Command or Modifier Command. For example, by specifying:

```
NAME-GEN S='ENTITY OR GROUP'
CONTENTS
```

in sequence, the CONTENTS REPORT will be generated for all ENTITY and GROUP names in the URA data base. In addition, the contents of the file produced by NAME-GEN are maintained until the next NAME-GEN is issued.

4.5 Using PUNCH Files

PUNCH files are files which have formats acceptable by FILE or INPUT parameters. The file described in the previous section is a PUNCH file from the NAME-GEN command. Output from NAME-GEN is put into its assigned PUNCH file so that it may be used as input to any of the FILE parameters for Modifier and Report Commands. The PUNCH file format is different from the report format for the command that generates both of them. For example, upon execution of the NAME-GEN command for all PROCESSES, the report

generated will consist of a report heading, line numbers for the contents of the report, the names of all PROCESSES in the data base and their corresponding name type (which is, of course, PROCESS). The contents of the PUNCH file produced by this command will only contain the names of the PROCESSES, without report headings, etc. In other words, the PUNCH file contains similar information to the report output from the command, but in a format acceptable to the FILE and INPUT parameters of other URA commands.

At most installations, the PUNCH file to be used (name of the file) can be specified by the PUNCH parameter for the command. The manner of assignment varies slightly from one installation to the next. (See Part II, Section 4.) Specific usage of the PUNCH parameter is given in the descriptions of the individual commands that utilize this parameter. The specific names of the PUNCH files used can be found in the appropriate Addendum for Appendix E.

5. RECEIVING OUTPUT FROM URA COMMANDS[1]

Several URA commands allow the user to specify whether output is generated from the command or not. This is done via the "output data" parameters for the particular command. When generating outputs from URA, the information is put into a file or printed on a device such as a line printer or terminal. If this file or device is not specified then all outputs are written to the main output file or device. This means that output will be written on the terminal when in conversational mode and on the line printer when running batch. There are several reasons why outputs might be routed elsewhere, expecially for on-line processing:

- Large quantities of output would take too long to be printed at the terminal.

- Depending on the type of terminal, some portions of the output may not be printed because of physical restrictions imposed by the terminal.

- The handling of printout from the terminal can sometimes be awkward and the format not aesthetically pleasing.

- No copy of the output is desired. (Only the PUNCH file may be needed as a step in a modification procedure.)

Most methods of receiving and controlling output from URA

---

[1] This section only deals with receiving outputs in the form of reports (as specified in Part III). Receiving output as presented via the PUNCH parameter is discussed in the previous section.

commands are installation dependent and therefore given in Part II, Section 5.

5.1 The NOPRINT Parameter

Several URA commands allow the option of not having the output printed via the NOPRINT parameter. The commands that allow this parameter are:

DELETE-COMMENT-ENTRY (DCOM)

FORMATTED-PROBLEM-STATEMENT (FPS)

NAME-GEN (NG)

PRINT-COMMENT-ENTRY (PCOM)

PROCESS-INPUT-OUTPUT (PRIO)

REPLACE-COMMENT-ENTRY (RCOM)

The two Modifier Commands, RCOM and DCOM, have this parameter available because the printout can be fairly large and may not be needed for future reference. The report heading for the RCOM or DCOM output and any error diagnostics are still printed to provide a hard-copy record of the command execution.

The remaining four Report Commands can use this parameter in conjunction with the PUNCH parameter. The option of the NOPRINT parameter is provided because when PUNCH information is desired, there may be no need for the printout.

5.2 The INDEX Parameter

Several commands allow the user to specify that an index (alphabetic listing of names used) for the report be generated by the command. The index also specifies the pages on which these names occur in the report. This is done by specifying INDEX as a parameter for the command. The commands which allow this parameter are:

CONTENTS
DICTIONARY
EXTENDED-PICTURE
FORMATTED-PROBLEM-STATEMENT
PICTURE
PROCESS-CHAIN
PROCESS-INPUT-OUTPUT
RESOURCE
SECURITY
STRUCTURE

5.3 The NOSOURCE and XREF Parameters

The NOSOURCE and XREF parameter have the same function for the INPUT-PSL and DELETE-PSL commands as the NOPRINT and INDEX parameter for other URA commands. The report produced by XREF, however, (the URA CROSS REFERENCE LISTING) specifies the lines rather than the pages where the names occur.

6. CONTROL COMMANDS

All control commands are installation-dependent. For this reason all control commands available to a particular installation and the descriptions and usage of these commands are given in Part II, Section 6.

7. MODIFIER COMMANDS

All the commands in this section modify the URA data base in some manner and generate an output to present the result of the modification. These outputs provide the user with a permanent record of changes made to the problem statement in the data base.

Effect of Modifier Commands on the Data Base Structure

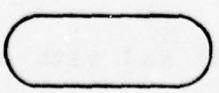
There are basically three types of information stored in a URA data base:

1) Name and types of objects defined by the user.

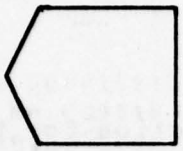
2) Comment entries (narrative and free format descriptions of objects).

3) Connections among objects and between an object and comment entry.

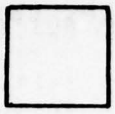
All this information is entered into the data base via INPUT-PSL from the URL statements used as input to this command. In most cases the section header statements define the type of objects and names of the objects, comment entry statements (DESCRIPTION, for example) define comment entries to be stored, and other URL statements define relationships or connections among the named objects in the data base. To present the structure of the data base in a graphical manner, the following symbols will be used:



- symbol for record used to describe a given named object (name record).

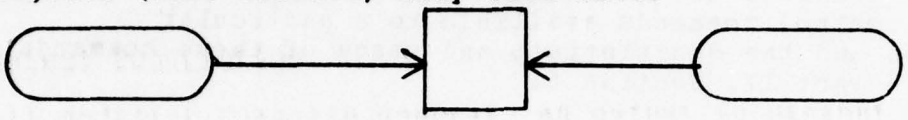


- symbol for comment entry stored in data base.



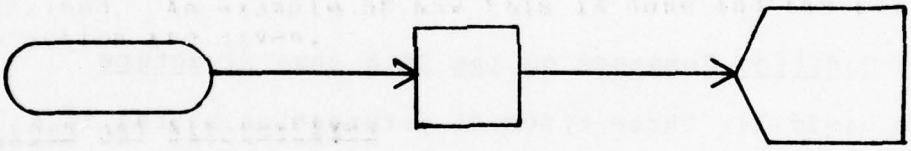
- symbol for a NUB, a type of record used to make connections among objects and between an object and comment entry.

Using the above notation, a simple relationship between two objects (name records) many look like:



Data in the NUB defines the type of connection (RECEIVES, for example) and the direction of the arrows defines the manner in which the relationship should be interpreted, i.e., which object does the RECEIVING.

A connection between an object and a comment entry may look like:



The data in the NUB defines the type of comment entry (PROCEDURE, for example).

It is important to note that the connections made among objects are different from the connection made between an object and comment entry.

INPUT-PSL creates records for named objects, the NUB records connecting the objects, and the comment entries. Commands must also be available to do the following:

1) Change a name record

   i) change the name of the object

   ii) change the object type

2) Delete a connection between objects

3) Delete a name record (and any connections it had with other objects)

4) Change a comment entry

5) Delete a comment entry

URA has facilities to perform all of these modifications on the data base information. The following URL commands perform the actions corresponding to the above:

1) Change a name record:

i) Change the name of an object:

RENAME - This command changes information within the name record only.

ii) Change the object type:

CHANGE-TYPE - This command changes information within the name record only.

2) Delete connections among objects:

DELETE-PSL - This command deletes NUBS (and thus connections) among name records in the data. It does not delete name records or comment entries.

3) Delete name records:

DELETE - This command deletes name records. Names to be deleted having connections with other names and/or having comment entries associated with them, will have corresponding NUBS (and comment entries) deleted.

4) Change comment entries:

REPLACE-COMMENT-ENTRY
- This command changes information within the comment entry only.

5) Delete comment-entries:

DELETE-COMMENT-ENTRY
- This command deletes comment entries and also deletes corresponding NUBS.

Modifier Command Description Format

Each Modifier Command will be described in the following format:

Command Name (command abbreviation)

When specifying the command to be executed, either the long form of the command name or legal abbreviations of the name may be given.

Modification Made

All Modifier Commands change information in the URA data base. What type of information is changed and how it is changed is described. Also, the checks made by URA before the change is made are presented.

Output Description

All Modifier Commands generate an output to present the result of the modification(s) made. The name of the output, purpose of the output, contents, and diagnostics given in the output are presented to aid in using it.

Execution

The basic form of specifying the command to be executed is described. An example of how this is done and the results from the action are given.

Options and Alternatives

All Modifier Commands can be executed in more than one way. For example, a different form of the command may be used to change one name in the data base versus a number of names. Also, the effect which the parameters for the command have on modifying the data is described. Examples are given to illustrate the alternatives.

Common Errors

It is possible to make particular errors in the use of each Modifier Command. Some of the particular logical and syntactical errors that occur when executing the command are given.

The following commands are described in this section in alphabetical order:

7.1 CHANGE-TYPE
7.2 DELETE
7.3 DELETE-COMMENT-ENTRY
7.4 DELETE-PSL
7.5 INPUT-PSL

7.6 PUNCH-COMMENT-ENTRY
7.7 RENAME
7.8 REPLACE-COMMENT-ENTRY

7.1 CHANGE-TYPE (CTL

Modification Made

Each name specified as input to this command has its corresponding name type changed if the new type does not conflict with the context in which the name has previously been used. This modification is most often used to change an underlined name (*** UNDEFINED ***) to a specific name type (such as GROUP or ELEMENT). Various "checking" facilities must be used to ascertain that legal changes are being made. For each name type change, URA must check to see that:

i) The name whose name type is to be changed exists in the data base.

ii) The assignment of the new name type is consistent with the context in which the name was used previously.

Output description

The output generated by this command is the CHANGE-TYPE REPORT. This report presents for each name used as input to the CHANGE-TYPE command, the name, the old name type associated with it and the new name type now assigned to it. Any error diagnostics which may occur during the name type change will also be printed. The names are printed out in the same order in which they were read as input to the CHANGE-TYPE command.

Execution

To change the name type of only one name, the following command format is issued:

```
CHANGE-TYPE NAME=gross-pay TYPE=GROUP
```

Previously in the problem statement, "gross-pay" had been defined as a ELEMENT. It was more appropriate to call it an GROUP, and the CHANGE-TYPE command made it easy to facilitate this change. The resulting CHANGE-TYPE REPORT is shown in Figure 2.

Options and Alternatives

1) The name types of several names can be changed at one time

if the names are put into a file and the file is specified as input to the command.[1] (This is usually done via the FILE parameter.) Figure 3 is an output resulting from using a file as input to CHANGE-TYPE. All the names in the example are shown to have been previously defined as ELEMENTS. The report shows that "birthdate" was changed to a GROUP, "number-of-deductions" became a GROUP, "surname" a GROUP and "gross-pay" changed back to an ELEMENT. Each line of the input file consists of the name of an object followed by the new name type to be assigned to it. The format is acceptable if the name is followed by its new name type and the two are within the first eighty columns of the file line and there is at least one blank between them. The file used to generate Figure 3 was:

```
birthdate GROUP
number-of-deductions GROUP
surname GROUP
gross-pay ELEMENT
```

2) The TYPE parameter can also be used effectively with an input file. All names in the input file will have their name types changed to the name type specified by the TYPE parameter. If the file from the previous example was specified as input and the TYPE parameter was also used, only the names in the file would be read as input; the name types would be ignored. All the names specified in the input file would have their name types changed to that given by the TYPE parameter. Figure 4 presents the output resulting from a CHANGE-TYPE command with TYPE=ELEMENT and the names used as input are the same as that used for Figure 3.

3) It is sometimes advantageous to use NAME-GEN in conjunction with CHANGE-TYPE, i.e., use the output produced by NAME-GEN as input to CHANGE-TYPE. This is most often done when all names of a particular type (usually UNDEFINED) are changed to another type (GROUPS or ELEMENTS). To change all undefined names in the data base to GROUPS:

```
NAME-GEN S='UNDEFINED'
CHANGE-TYPE FILE TYPE=GROUP
```

The NAME-GEN command selects all undefined names and places them in a file. The CHANGE-TYPE command then uses the file produced by NAME-GEN as its input file. The TYPE=GROUP parameter specifies that all the names in the input file should be changed to GROUPS. Figure 5 presents the result of this procedure.

---

[1] That exact manner in which the file is specified is given in Part II, Section 7.1.

FIGURE_2

CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

NAME=gross-pay TYPE=GROUP

1* gross-pay
OLD TYPE - ELEMENT
NEW TYPE - GROUP

FIGURE_3

CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

FILE

1* birthdate
OLD TYPE - ELEMENT
NEW TYPE - GROUP

2* number-of-deductions
OLD TYPE - ELEMENT
NEW TYPE - GROUP

3* surname
OLD TYPE - ELEMENT
NEW TYPE - GROUP

4* gross-pay
OLD TYPE - GROUP
NEW TYPE - ELEMENT

FIGURE_4

CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

FILE TYPE=ELEMENT

1* birthdate
OLD TYPE - GROUP
NEW TYPE - ELEMENT

2* number-of-deductions
OLD TYPE - GROUP
NEW TYPE - ELEMENT

3* surname
OLD TYPE - GROUP
NEW TYPE - ELEMENT

4* gross-pay
OLD TYPE - ELEMENT
NEW TYPE - ELEMENT

FIGURE_5

CHANGE-TYPE REPORT

```
PARAMETERS FOR: CHANGE-TYPE

   FILE TYPE=GROUP

 1*  error-flags
   OLD TYPE - *** UNDEFINED ***
   NEW TYPE - GROUP


 2*  error-messages
   OLD TYPE - *** UNDEFINED ***
   NEW TYPE - GROUP
```

Common Errors

If neither an input file nor NAME is specified for the command, the message: "NO NAME GIVEN" will be printed by URA. If a file or NAME is given, but no name types are given in the file or no TYPE is given, the message: "NO TYPE GIVEN WITH "NAME=" OR "FILE" PARAMETER will be printed. .Should either of these messages be generated, URA will not execute the CHANGE-TYPE command. The command and its parameters should be re-entered with the necessary corrections. Another common error is attempting to assign a new name type to a name that has previously been used in a way that conflicts with its new name type. For example, if the name XYZ was previously defined to be CONTAINED in SET S1, this would imply that XYZ must be either an ENTITY, INPUT or OUTPUT. If an attempt was made to change its type to a GROUP (which cannot be CONTAINED in a SET) the error message:

URA029:MAINCT: CONFLICT WITH EXISTING CONNECTIONS

would be given and the change would not be made. If it is still desired that XYZ be a GROUP, XYZ should be deleted from the data base and proper URL statements for XYZ should be entered via INPUT-PSL. All conflicting connections are listed. They all must be resolved before the change of type can occur.

7.2 Delete (DEL)

Modification Made

For each name specified as input to the DELETE command, all its relationships (i.e., USES, SUBPARTS, etc.), with other names in the data base are removed, its comment entries (such as DESCRIPTION or PROCEDURE) are deleted and finally the name is deleted from the data base. Before any of these modifications are made, URA checks that the name to be deleted exists in the data base. If the name cannot be found, no attempt will be made by URA to delete the name.

Output Description

The DELETION REPORT is produced each time this command is initiated. Each name used as input to the DELETE command is printed on the report along with the status of the change (i.e., if it did or did not work). The names on the output appear in the same order as read by the DELETE command.

This report serves as a permanent record of names that have been deleted from the URA data base. It is intended to aid the analyst in keeping track of modifications to the data base.

Once there is a record of a particular name being deleted, the analyst has the option of re-using the name.

Execution

The following command deletes one name from the data base:

DELETE NAME=remaining-funds

The DELETION REPORT for this action is shown in Figure 6.

Options and Alternatives

1) Several names can be deleted from the data base if the names are put into a file and the file is specified as input to the command.[1] (This is usually done via the FILE parameter.) The format of the input file consists of one name per line, beginning in column one. Figure 7 is an output resulting from using a file as input to the DELETE command. All the names in the report had been defined in the data base. The report shows that deletion of each name was successful.

Common Errors

DELETE should not be used to delete the entire contents of the data base. Operating system command should be utilized for this procedure. The data base should be emptied and then reinitialized.

When doing minor editing of a URL description for a name in the data base the information connected to that name (URL statements) should be saved before deleting the name. A FORMATTED-PROBLEM-STATEMENT for the name should be generated using the PUNCH parameter. Then the name can be deleted (via DELETE) from the data base. At this point, the old information in the PUNCH file can be edited to suit the problem definer and then re-entered via the INPUT-PSL command using the PUNCH file produced by the PPS as input.

If a file is used as input to the command, all names to be deleted must begin in the first column of the file line. Any preceding blanks will be interpreted as part of the name.

If neither an input file nor NAME is specified for the command, the message: "NO NAME OR FILE WAS SPECIFIED" will be printed by URA. Should this happen, URA will not execute the DELETE command. The command and its parameters should be reentered

---

[1] The exact manner in which the file is specified is given in Part II, Section 7.2.

FIGURE_6

D E L E T I O N

```
PARAMETERS FOR: DEL

   NAME=remaining-funds

DELETED - remaining-funds
```

FIGURE_7

D E L E T I O N

```
PARAMETERS FOR: DEL

   FILE

DELETED - error-listing
DELETED - hired-employee-report
DELETED - hourly-employee-report
DELETED - pay-statement
DELETED - paysystem-outputs
DELETED - salaried-employee-report
DELETED - terminated-employee-report
```

with the necessary corrections.

7.3 DELETE-COMMENT-ENTRY (DCOM)

Modification Made

The DELETE-COMMENT-ENTRY takes those names specified as input and deletes, for each input name, the comment entries associated with those comment entry types designated as command parameters.[1] If no comment entry types are specified by the parameters, no comment entries will be deleted. Checking is performed to see if the comment entry exists in the data base before it is deleted. If the comment entry cannot be found, no attempt is made to delete it.

Output Description

The output report generated by this command is called DELETED COMMENT ENTRIES. For each comment entry to be deleted from the data base, the following information is printed on the output:

- name in the data base to which the comment entry belonged.

- the type of comment entry (i.e., DESCRIPTION, PROCEDURE, etc.) which is being deleted.

- the full text of the comment entry.

The order of the output names is the same as the order of the input names.

This serves as a hard copy record for those comment entries deleted from the system description. As stated before, it is desirable to have all modifications to the system description documented.

Execution

The following command deletes the PROCEDURE comment entry associated with one name.

```
DCOM NAME=employee-processing PROCEDURE
```

This was done because the problem definers wanted to delete current PROCEDURE comment entry and did not want to replace it.

---

[1] An example of a comment entry type is a URL "DESCRIPTION" or "PROCEDURE" statement. The comment entry associated with this comment entry type would be the text specified by the user

If the comment entry could be correct if edited or a replacement was available, then it would be more appropriate to use the REPLACE-COMMENT-ENTRY command (see section 7.8). The output generated by this command is shown in Figure 8.

Options and Alternatives

1) Multiple comment entries can be deleted for a name:

```
DCOM NAME=new-info-validation PROCEDURE DESCRIPTION
```

In this case, the PROCEDURE and DESCRIPTION comment entries would be deleted for the PROCESS, "new-info-validation."

2) The following types of comment entries can be deleted when specified as parameters for DELETE-COMMENT-ENTRY:

```
DERIVATION
DESCRIPTION
FALSE-WHILE
PROCEDURE
TRUE-WHILE
VOLATILITY
VOLATILITY-MEMBER
VOLATILITY-SET
```

3) Several names can have their comment entries deleted if the names are put into a file and the file is specified as input. The comment entries deleted for these names are those specified as parameters for the command. If the user specifies that the DESCRIPTION and PROCEDURE comment entries be deleted for a GROUP name, the DESCRIPTION comment entry will be deleted and the message:

```
URA104:DELSET: PROCEDURE COMMENT ENTRY NOT FOUND
```

will be given because GROUPS may not have PROCEDURE statements.

Figure 9 shows the output resulting from executing the DELETE-COMMENT-ENTRY command with a file of names as input and DESCRIPTION and PROCEDURE comment entry types as parameters. The example shows for each name used as input, each comment entry deleted for the name.

4) Printing of the DELETED COMMENT ENTRIES output may be suppressed by specifying NOPRINT as a parameter:

```
DCOM NAME=payroll-processing DESCRIPTION NOPRINT
```

Common Errors

FIGURE_8

D E L E T E D C O M M E N T E N T R I E S

```
PARAMETERS FOR: DCOM

   NAME=new-employee-processing NODESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER
   NOVOLATILITY-SET NODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT NOFILE


1*   new-employee-processing
           PROCEDURE;
        1             1. add new employee information
        2             2. increment count of number of employees in appropriate
        3                department
        4             3. specify relationship between employee information and
        5                department
        6             4. initialize all appropriate fields in employee information. ;
```

FIGURE_9

D E L E T E D C O M M E N T E N T R I E S

```
PARAMETERS FOR: DCOM

   DESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
   NOTRUE-WHILE NOFALSE-WHILE PRINT FILE


 1*   hourly-employee-processing
            DESCRIPTION;
      1             This process performs those actions needed to interpret
      2             time cards to produce a pay statement for each hourly
      3             employee. ;


 2*   hourly-employee-processing
            PROCEDURE;
      1             1. compute gross pay from time card .
      2             2. compute tax from gross pay.
      3             3. subtract tax from gross pay to obtain net pay.
      4             4. update hourly employee information accordingly.
      5             5. update department information accordingly.
      6             6. generate paycheck.
      7            Note: if status code specifies that the employee did not work
      8              this week, no processing will be done for this employee
      9              information. ;


 3*   salaried-employee-processing
            DESCRIPTION;
      1             This process produces the pay statement for salaried
      2             employees once a month. ;


 4*   salaried-employee-processing
            PROCEDURE;
      1             1. salary defines gross pay.
      2             2. compute taxes from gross pay.
      3             3. subtract taxes from gross pay to obtain net pay.
      4             4. update salaried employee information accordingly.
      5             5. update department information accordingly.
      6             6. generate paycheck
      7            Note: hours worked is assumed to be 40 ;
```

FIGURE_9

D E L E T E D C O M M E N T E N T R I E S

```
5*   process-library
           DESCRIPTION;
      1            These are routines used by one or more processes
      2            in the system. ;
```

If neither an input file nor NAME is specified for the command, the message "NONAME OR FILE SPECIFIED" will be printed by URA. Should this happen, URA will not execute the DELETE-COMMENT-ENTRY command. The command and its parameters should be reentered with the necessary corrections.

7.4 DELETE-PSL (DPSL)

Modification Made

This command takes as input, any URL statements in the format specified in the "User Requirements Language, Version 3.2 Language Reference Manual."[1] For each section header statement (i.e., PROCESS, DEFINE, etc.)[2] all the URL statements following this section header (up to the next section header statement) will be deleted from the URA data base for those names specified in the header statement. This command only deletes relationships between names and does not delete any comment entries (this is handled by the DCOM command) nor deletes names (this is handled by the DELETE command). If some of the information presented by the URL statements is contradictory, an error message will be given for that statement. Error diagnostics are also given when syntactical errors occur. URA attempts to continue the procedure until too many errors are encountered.[3]

Output Description

The two outputs that may be generated by this command are DELETED URL and the URA CROSS REFERENCE. The DELETED URL output is a record of all information (except names and comment entries) deleted from the URA data base. It aids the analyst in finding errors in the deletion procedure and produces error diagnostics in sufficient detail to aid the analyst in correcting these errors. This output displays, line for line, the data used as input to the DELETE-PSL command. No reordering is done on the input data.

The URA CROSS REFERENCE is intended as an aid to the analyst in correcting errors that appear in the DELETED URL output. It consists of an alphabetical list of all user defined names, i.e., non-URL names that appear in the DELETED URL output. For each name that appears in the CROSS REFERENCE, its corresponding name type (as given in the DELETED URL output) is printed and a list of all lines in the DELETED URL output where the name

---

[1] Part II of the URL User's Manual.
[2] See Appendix F of the URL User's Manual. List of all possible section header types.
[3] See Section 9 for the limit of errors allowed.

appeared is also given.

Execution

The most common method of deleting URL statements is by first writing all statements to be deleted into a file (or punching them on cards) and then using this as input to the command.[1] (This is usually done via the INPUT parameter.) Only the first 72 columns of each line in the file may contain URL statements. Anything after column 72 is ignored. Figure 10 is the output resulting from this type of procedure. The EOF statement must be given to specify the end of URL statements to be deleted.

Options and Alternatives

1) In many cases the amount of input is relatively large (a few hundred lines); hence, there may be a need for the URA CROSS REFERENCE. It will be generated by specifying the XREF parameter with the command.

2) If no input file is specified, URA will wait for URL statements to be typed in (from the terminal), or when in batch mode, interpret any following cards up through the first "EOF" as URL statements to be deleted. When URL statements are entered at the terminal, each line entered is echoed back by URA along with any errors encountered for that statement (i.e., an AS-IS SOURCE LISTING). This allows the user to correct errors as they occur.

3) Printing of the DELETED URL output may be suppressed by specifying NOSOURCE as a parameter.

Common Errors

The most common errors are typing errors encountered in interpreting the URL statements. A typing mistake can cause many different types of syntactical and semantic errors.

Only the first 72 columns of each line in the input file are read so all URL statements should fit in this region. Anything over column 72 will be truncated and an error message will be generated in most cases.

Omitting the semicolon at the end of a URL statement is a common cause for several errors.

DELETE-PSL will not delete comment entry statements from the

---

[1] The exact manner in which the file can be specified is given in Part II, Section 7.4.

FIGURE_10

DELETED URL

PARAMETERS FOR: DPSL

SOURCE NOXREF

```
LINE  S T M T                                                    ID FIELD

   1 >GROUP: check;
   2 >  CONTAINED: pay-statement;
   3 >RELATION: comp-pay-info;
   4 >OUTPUT: paycheck;
   5 >  CONSISTS: pay-stub;
   6 >  RECEIVED BY: employee;
   7 >EOF
```

data base so these statements are ignored if encountered in the input file. No names other than SYNONYMS can be deleted from the data base via DELETE-PSL.

The last line of the input file containing the URL statements should have the word EOF signifying the end of input. This should also be typed when inputting the data interactively. EOF allows the return to the URA command handler. No URL statements are read after EOF.

7.5 INPUT-PSL (IP)

Modification Made

This command takes as input, any URL statements in the format specified in "User Requirements Language, Version 3.2, Language Reference Manual."[1] For each section header statement (i.e., PROCESS, DEFINE, etc.)[2] the user defined names specified by that section header will be added to the list of names in the data base (if not already in the data base). All the URL statements following this section header up to the next section header statement, specify connections to be made with other names in the data base. URA first performs syntax and semantic checks on each input line before any more complex checking is performed. An "in context" check is made for each name used as input. If the name is not in the user's data base, it is added. If it is, a check is made to see that the context in which the name is used in the new input agrees with the manner in which the name is used in the data base. If there is a conflict, an error message will be produced and URA will skip to the next input statement. URA attempts to continue the input procedure until too many errors are encountered.[3] If redundant information is given, i.e., specifying the same relationship more than once, the redundant information will not be added to the information already in the data base. No diagnostic message is given to denote redundant information.

Output Description

The two outputs that may be generated by this command are the URA AS-IS SOURCE LISTING and the URA CROSS REFERENCE. The URA AS-IS SOURCE LISTING is a record of all information input into the URA data base, and is intended as an aid to the analyst. It

---

[1] Part II of the URL User's Manual.
[2] See Appendix F of the URL User's Manual. All possible section header types.

---

[3] See Section 9 for the limit of errors allowed.

aids the analyst in finding errors in the input data and produces error diagnostics in sufficient detail to aid the analyst in correcting these errors. The output displays, line for line, the data used as input to the INPUT-PSL command. The order of the input data is not changed.

The URA CROSS REFERENCE is intended as an aid to the analyst in correcting errors that appear in the URA AS-IS SOURCE LISTING and also to resolve ambiguities in assigning name types to the undefined names in the listing. It is useful in correcting errors, as any name involved in an error can be quickly referenced to find all places in the AS-IS LISTING where the name is used, and the name type assigned to that name.

From this information, the analyst will be able to determine what information has to be reentered to correct the error. Since the CROSS REFERENCE also presents all those names which have an ambiguous name type (one that was not defined in previous input), the analyst can resolve those ambiguities by use of the CHANGE-TYPE or INPUT-PSL commands. The output consists of an alphabetical list of all user defined names, i.e., non-URL names, that appear in the AS-IS LISTING. For each name that appears in the CROSS REFERENCE, its corresponding name type (as given in the AS-IS LISTING) is printed and a list of all lines in the AS-IS LISTING where the name appeared is also given.

Execution

The most common method of inputting URL statements is by first writing all statements to be added into a file (or punching them on cards) and then using this as input to the command.[1] (This is usually done via the INPUT parameter.) Note that only the first 72 columns of each line in the file may contain URL statements. Anything after 72 is ignored. Figure 11 is the output resulting from this type of procedure. The EOF statement must be given to specify the end of URL statements to be added. The UPDATE parameter specifies that the URA data base is to be modified by the input. If this parameter is not given, none of the information will be added to the data base.

Options and Alternatives

1) In many cases, when the amount of input is relatively large (a few hundred lines) there may be a need for the URA CROSS REFERENCE. By simply specifying XREF as a parameter, it will be generated. Figure 12 presents an AS-IS LISTING and CROSS REFERENCE for a small problem statement.

---

[1] The exact manner in which the file can be specified is given in Part II, Section 7.5.

FIGURE_11

A S - I S S O U R C E L I S T I N G

PARAMETERS FOR: SYNU

SOURCE NOXREF NOUPDATE DBREF

```
LINE  S T M T                                                    ID FIELD

    1 >/* Communication Aids 8 */
    2 >
    3 >INPUT: employee-information;
    4 >     SYNONYM: emp-info,i1;
    5 >     DESCRIPTION;
    6 >          This input represents all the necessary information to
    7 >          produce the outputs from the paysystem. ;
    8 >
    9 >OUTPUT: paysystem-outputs;
   10 >     SYNONYM: payouts,o1;
   11 >     DESCRIPTION;
   12 >          This output represents all the required outputs of the
   13 >          target paysystem as defined by policy. ;
   14 >
   15 >SET: payroll-master-information;
   16 >     SYNONYM: pay-mast,master-file,s1;
   17 >     DESCRIPTION;
   18 >          This set contains one unit of information
   19 >          for each employee on the payroll, that is,
   20 >          those employees who are to receive paychecks.;
   21 >
   22 >INTERFACE: departments-and-employees;
   23 >     SYNONYM: dept-emp,r1;
   24 >     DESCRIPTION;
   25 >          This is the entity which will receive all the outputs and
   26 >          supply all the inputs. ;
   27 >
   28 >PROCESS: payroll-processing;
   29 >     SYNONYM: payproc,p1;
   30 >     DESCRIPTION;
   31 >          This process represents the highest level process
   32 >          in the target system.  it accepts and processes
   33 >          all inputs and produces all outputs.;
   34 >
   35 >EOF
```

28a

FIGURE_12

A S - I S S O U R C E L I S T I N G

PARAMETERS FOR: SYNU

SOURCE XREF UPDATE DBREF

```
LINE  S T M T                                                    ID FIELD
   1 >/* System Flow 8 */
   2 >
   3 >INPUT: employee-information;
   4 >
   5 >OUTPUT: paysystem-outputs;
   6 >
   7 >SPT: payroll-master-information;
   8 >
   9 >INTERFACE: departments-and-employees;
  10 >      GENERATES: employee-information;
  11 >      RECEIVES: paysystem-outputs;
  12 >
  13 >PROCESS: payroll-processing;
  14 >      UPDATES payroll-master-information;
  15 >      RECEIVES: employee-information;
  16 >      GENERATES: paysystem-outputs;
  17 >
  18 >EOF
```

FIGURE_12

C R O S S R E F E R E N C E

SEQ	NAME	TYPE	
1	departments-and-employees	INTERFACE	
	9		
2	employee-information	INPUT	
	3	10	15
3	payroll-master-information	SET	
	7	14	
4	payroll-processing	PROCESS	
	13		
5	paysystem-outputs	OUTPUT	
	5	11	16

2) In most cases, it is advantageous to first do a syntax and semantic check of the input data before attempting to put it in the data base. By not specifying the UPDATE parameter, these checks will be made without actually putting the information into the data base. This will generate an AS-IS LISTING with error diagnostics for the URL statements used as input. Since most problem statements have one or two typing errors anyway, this proves to be an inexpensive way to catch errors early. After the source of the errors has been determined and corrected, the command can be issued again using UPDATE as a parameter.

3) If no input file is specified, URA will wait for URL statements to be typed in (from the terminal), or when in batch mode, interpret any following cards up through the first "EOF" as URL statements to be added to the data base. When URL statements are entered at the terminal, each line entered is echoed back by URA along with any errors encountered for that statement (i.e., an AS-IS SOURCE LISTING). This allows the user to correct errors as they occur.

4) Printing of the AS-IS SOURCE LISTING may be suppressed by specifying NOSOURCE as a parameter.

5) The DBREF parameter allows referencing of the data base when semantic as well as syntax checks are desired for URL statements used as input. This must be in effect when UPDATE is given as a parameter. NODBREF may be specified if only a syntax check of the statements is desired and the data base is not to be updated.

Common Errors

The most common errors are typing errors encountered in interpreting the URL statements. A typing mistake can cause so many different types of syntactical and semantic errors that it will be handled in a later section (section 10).

It is also possible to input the new information into the wrong data base. It is important that the data base to be used has been specified. Otherwise, the data base may be set to some default which may not be the data base file desired.

The INPUT-PSL command only reads the first 72 columns of each line in the input file so all URL statements must fit in this region. Anything over column 72 will be truncated and an error message will be given in most cases.

Omitting the semicolon at the end of a URL statement is a common cause for several errors. It is important that the syntax of

each URL statement be correct.

The last line of the input file containing the URL statements should have the word EOF signifying the end of input. This should also be typed when inputting the data interactively. EOF allows the return to the URA command handler. (See Figures 11 and 12 to see how EOF is used correctly.) No URL statements are read after EOF.

7.6 PUNCH-COMMENT-ENTRY (PCOM)

Modification Made

Technically, the output produced by this command is a report presenting narrative information in the manner of a glossary. It makes no modifications to the data base, but is presented here because its main objective is to aid the analyst in changing comment entries in conjunction with the REPLACE-COMMENT-ENTRY command. The idea of using the output as a glossary (for final specifications perhaps) however, should not be overlooked. A message is given when no comment entry is available for a particular comment entry type or the name specified is not in the data base.

Output Description

The PUNCHED COMMENT ENTRIES output is generated by this command. It presents selected comment entries for each name used as input to the command. Any type of name may be used as input to the command. Depending on the type of name the following comment entries may be retrieved:

DERIVATION	(DER)
DESCRIPTION	(DESC)
FALSE-WHILE	(FW)
PROCEDURE	(PRCD)
TRUE-WHILE	(TW)
VOLATILITY	(VOL)
VOLATILITY-MEMBER	(VOLM)
VOLATILITY-SET	(VOLS)

For each name used as input to the command, the name is printed on the output in the order in which it was read and associated with that name, the type of comment entry and the text for that comment entry (for each type of comment entry as specified in the parameter list).

Execution

To obtain the DESCRIPTION comment entry for one name the

following command might be given:

```
PCOM NAME=payroll-processing DESCRIPTION
```

This will generate the report shown in Figure 13. A PUNCH file will also be generated with the same information as the report. The manner in which the file to contain the PUNCH data is specified is installation dependent and given in Part II, Section 7.6. If the procedure is done at the terminal, the PUNCH file can then be edited and used as input to the REPLACE-COMMENT-ENTRY command to modify the comment entry. If the procedure is done in batch, the contents of the PUNCH file produced should be punched on cards (by the system if possible). Then the deck of cards produced can be modified and used as input to REPLACE-COMMENT-ENTRY in the next batch run.

Options and Alternatives

1) Several names can have comment entries printed and/or PUNCHED if the names are put into a file and the file is specified as input to the command.[1] (This is usually done via the FILE parameter.) Figure 14 is an output resulting from using a file as input to the PCOM command.

2) Multiple comment entries, such as DESCRIPTION and PROCEDURE, can be generated for several names when a file is specified as input and more than one comment entry type is specified as parameters. This is illustrated in Figure 15.

3) When the objective of executing this command is to generate a PUNCH file, printing of the report may be suppressed by issuing NOPRINT as a parameter.

4) When the objective of executing this command is to generate the report (and no PUNCH data is desired), production of data in the PUNCH file may be suppressed by issuing NOPUNCH as a parameter.

5) The NAME-GEN can also be used in conjunction with PCOM to retrieve all names of a particular name type (such as INTERFACE) to be used as input to the PCOM command. For example:

```
NAME-GEN S='INTERFACE'
 PCOM     DESCRIPTION
```

This procedure retrieves all INTERFACE names defined in the data base and produces the PUNCHED COMMENT ENTRIES report

---

[1] The exact manner in which the file is specified is given in Part II, Section 7.6.

FIGURE_13

PUNCHED COMMENT ENTRIES

```
PARAMETERS FOR: PCOM

   NAME=payroll-processing DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER
   NOVOLATILITY-SET NODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH


1*   payroll-processing
     DESCRIPTION;
          1              This process represents the highest level process
          2              in the target system.  it accepts and processes
          3              all inputs and produces all outputs.;
```

FIGURE_14

PUNCHED COMMENT ENTRIES

```
PARAMETERS FOR: PCOM

   FILE DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
   NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH


 1*   employee-information
      DESCRIPTION;
           1              This input represents all the necessary information to
           2              produce the outputs from the paysystem.;

 2*   employment-termination-form
      DESCRIPTION;
           1              This input contains the information necessary to
           2              delete an employee from the payroll.;

 3*   hourly-employment-form
      DESCRIPTION;
           1              This input contains the information necessary to
           2              add a new hourly employee to the payroll.;

 4*   salaried-employment-form
      DESCRIPTION;
           1              This input contains the information necessary to
           2              add a new salaried employee to the payroll.;

 5*   tax-withholding-certificate
      DESCRIPTION;
           1             This input contains tax information necessary to
           2             compute the employee's paycheck.;

 6*   time-card
      DESCRIPTION;
           1              This input contains the information about the hours that an
           2              hourly employee worked the preceding week;
```

FIGURE_15

PUNCHED COMMENT ENTRIES

```
PARAMETERS FOR: PCOM

   FILE DESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
   NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH


 1*   hourly-employee-processing
      DESCRIPTION;
           1              This process performs those actions needed to interpret
           2              time cards to produce a pay statement for each hourly
           3              employee.;

 2*   hourly-employee-processing
      PROCEDURE;
           1              1. compute gross pay from time card .
           2              2. compute tax from gross pay.
           3              3. subtract tax from gross pay to obtain net pay.
           4              4. update hourly employee information accordingly.
           5              5. update department information accordingly.
           6              6. generate paycheck.
           7             Note: if status code specifies that the employee did not work
           8               this week, no processing will be done for this employee
           9               information.;

 3*   new-employee-processing
      DESCRIPTION;
           1              This process stores information about new employees and
           2              then prints out a corresponding report.;

 4*   new-employee-processing
      PROCEDURE;
           1              1. add new employee information
           2              2. increment count of number of employees in appropriate
           3                 department
           4              3. specify relationship between employee information and
           5                 department
           6              4. initialize all appropriate fields in employee information.;

 5*   salaried-employee-processing
      DESCRIPTION;
           1              This process produces the pay statement for salaried
```

FIGURE_15

PUNCHED COMMENT ENTRIES

```
     2              employees once a month.;

6*   salaried-employee-processing
     PROCEDURE;
     1              1. salary defines gross pay.
     2              2. compute taxes from gross pay.
     3              3. subtract taxes from gross pay to obtain net pay.
     4              4. update salaried employee information accordingly.
     5              5. update department information accordingly.
     6              6. generate paycheck
     7             Note: hours worked is assumed to be 40;

7*   terminating-emp-processing
     DESCRIPTION;
     1              This process deletes data, for those employees who
     2              are no longer on the payroll, from the files. It also
     3              prints a list of all employees no longer on the
     4              payroll.;

8*   terminating-emp-processing
     PROCEDURE;
     1              1. determine type of employee by employment status item
     2              2. from this, retrieve the contents of the appropriate
     3                 employee information and print in report format
     4              3. update number of employees field in appropriate
     5                 department information
     6              4. delete employee information.;
```

for all these names and their corresponding DESCRIPTIONS. This could also be done for more than one type of name:

```
NAME-GEN S='SET OR PROCESS'
 PCOM DESCRIPTION PROCEDURE
```

Notice that the PROCEDURE parameter is given, but SETS cannot have PROCEDURE statements associated with them. Only the DESCRIPTION statements will be retrieved for SET names while both DESCRIPTION and PROCEDURE statements will be retrieved for PROCESS names.

Common Errors

The problem definer should note that most of the parameters indicating comment types (i.e., FALSE-WHILE, VOLATILITY, etc.) can apply to only one type of name (CONDITION, ENTITY, respectively).

7.7 RENAME (REN)

Modification Made

The RENAME command takes an old name (of some object in the problem statement data base) and a new name as input. If the new name is not a URL reserved word or a name already in the data base, the command will replace the old name by the new name. Before a name is changed, URA checks that:

- the old name exists in the data base.
- the new name is not already used in the data base.
- the new name is a legal URL name (see the "User Requirements Language, Version 3.2 Language Reference Manual").[1]

If any of these requirements are violated, no change will be made.

Output Description

The output generated from this command is called the RENAME REPORT. For every name changed by the RENAME command, this report presents the "old name" which appeared in the data base and the "new name" which has taken its place. When the name change is not successful, error diagnostics are also printed specifying the cause of the error. Again, the names are printed

---

[1] Part II, URL Manual

on the output in the same order as they are read as input.

Execution

To change one name in the data base, the following command might be given:

```
RENAME OLD=employee-identification-number NEW=employee-id
```

Upon first defining the target system, "employee-identification-number" was used to represent a certain piece of data. Later it was found that this data was actually called "employee-id" and the change was made to be consistent with organization terminology. See Figure 16 for the report generated by this command. This command is also beneficial for changing misspelled names in the data base. Through typing errors, "employee-number" may have gone in as "emplyee-nuber". This mistake can be corrected by:

```
RENAME OLD=emplyee-nuber NEW=employee-number
```

Options and Alternatives

As with most of the modifier commands, the problem definer has the option of changing several names at one time. The old-new name pairs must first be put in a file to be used as input to the command. (This is usually done via the INPUT parameter.) Figure 17 presents the output resulting from this procedure. Each line of the file must consist of an old name followed by the corresponding new name. The two names may be anywhere in the first 80 columns of the line and must be separated by one or more blanks. The format of the file used to produce Figure 17 is given below:

```
employee-information        paysystem-inputs

employee                    personnel
```

Common Errors

The most common error in using RENAME is specifying a name already in the data base or a URL reserved word as the new name. URA will not make the change if this is the case. The command would have to be reissued with another new name to take the place of the illegal one.

If neither an input file or an OLD/NEW pair of parameters is specified for the command, the message: "MUST GIVE OLD AND NEW, OR INPUT" will be printed by URA. Should this happen, URA will not execute the RENAME command. The command and its parameters should be reentered with the necessary corrections.

FIGURE_16

RENAME REPORT

PARAMETERS FOR: REN

OLD=employee-identification-number NEW=employee-id

SEQ	OLD NAME	NEW NAME
1	employee-identification-number	employee-id

FIGURE_17

RENAME REPORT

PARAMETERS FOR: REN

INPUT

SEQ	OLD NAME	NEW NAME
1	employee-information	paysystem-inputs
2	employee	personnel

7.8 REPLACE-COMMENT-ENTRY (RCOM)

Modification Made

This command takes as input names which exist in the data base, each followed by a URL comment entry statement. If the comment is a DESCRIPTION comment entry, for example, then the command will replace the old DESCRIPTION comment entry by the one used as input. What RCOM actually does is delete the old comment entry and put the new comment entry in its place. This is done after a check has been made to ascertain that the "old comment entry" exists in the data base and the "new comment entry" is legal for the particular application being used (e.g., not attempting to enter a PROCEDURE comment entry for a SET name). A check is made to see if the input name is in the data base.

If an attempt is made to replace a comment entry for a name that did not have a comment entry specified for it, the message:

URA 126:REPSET: WARNING - THERE IS NO COMMENT ENTRY TO DELETE

will be given and the designated comment entry will be connected to the name.

Output Description

The output generated by this report is called REPLACED COMMENT ENTRIES. For each "old comment entry" to be replaced, the output depicts, in the following order:

- name to which the "old comment entry" belongs.
- the type of comment entry which is being changed.
- the entire text of the "old comment entry."
- the entire text of the "new comment entry" which replaces the old one.

Error diagnostics referring to problems encountered in executing the command are also printed here.

Execution

Any information to be supplied as input to RCOM must first be

---

[1] The exact manner in which the file is specified is given in Part II, Section 7.8.

placed in a file and the file must be designated as input.[1] (This is usually done via the INPUT parameter.)

The contents of the file must be in the following format:

```
name
comment entry type;
.
.
.
comment entry
.
.              ;
.
etc.
```

The contents of the file used to produce the output shown in Figure 18 was:

new-employee-processing DESCRIPTION; This process produces the new hire section in the h-t report.; new-employee-processing PROCEDURE; 1. add new employee information 2. increment count of number of employees in appropriate department 3. specify relationship between employee information and department 4. Initialize all appropriate fields in employee information. 5. Print the new hire section of the h-t report.;

Options and Alternatives

1) It is often the case that only some minor editing of a comment entry need be done to make it correct. This can be done when the output from the PUNCH-COMMENT-ENTRY command is edited and used as input to the RCOM command. This is described in section 7.6 of this paper.

2) To suppress printing of the REPLACED COMMENT ENTRIES report the NOPRINT parameter may be specified.

Common Errors

The major problem in using this command is specifying the file format correctly. (This is not a problem, however, if the contents of the file used was produced by PUNCH-COMMENT-ENTRY.) Although the command allows free formatting of the file, the order: name, comment-entry type, comment-entry must be maintained. Each must begin on a new line.

8. REPORT COMMANDS

Report Commands retrieve specific types of information from the

FIGURE_13

REPLACED COMMENT ENTRIES

```
PARAMETERS FOR: RCOM

   PRINT



** DELETED COMMENT ENTRY **
  1*   new-employee-processing
                DESCRIPTION ;
           1              This process stores information about new employees and
           2              then prints out a corresponding report.;

** INSERTED COMMENT ENTRY **
  1*   new-employee-processing
                DESCRIPTION ;
           1              This process produces the new hire section
           2              in the h-t report.;


** DELETED COMMENT ENTRY **
  2*   new-employee-processing
                PROCEDURE ;
           1              1. add new employee information
           2              2. increment count of number of employees in appropriate
           3                 department
           4              3. specify relationship between employee information and
           5                 department
           6              4. initialize all appropriate fields in employee information.;

* INSERTED COMMENT ENTRY **
  2*   new-employee-processing
                PROCEDURE ;
           1              1. add new employee information
           2              2. increment count of number of employees in appropriate
           3                 department
           4              3. specify relationship between employee information and
           5                 department
           6              4. initialize all appropriate fields in employee information.
           7              5. print the new hire section of the h-t report.;
```

data base and present it in formats which aid the problem definer to analyze the problem statement for correctness and completeness. Many of the formats can serve as final specifications of the system being designed.

Most report commands allow the report to be generated for a single name (via the NAME parameter) or for a number of names placed in a file and specified as input to the command.[1]

The descriptions of these report commands, their usage and interpretation, and the usage of reports produced by them are given in "URA Outputs."[2]

9. ERROR DIAGNOSTICS

URA has extensive checking facilities to prevent errors in the problem statement. At the URA command mode level, checks are made to insure that all commands given are legal URA commands and that all parameters given are legal parameters for that command. If an illegal command, an illegal parameter, or illegal parameter for that command is given when in on-line mode, the following message will be generated:

```
INVALID PARAMETER -
ENTER REPLACEMENT OR BLANK LINE
?
```

The user must enter the replacement following the question mark and then hit the carriage return key. If the command is accepted, processing of that command commences. Should an error be encountered while processing the command, one of the following three types of error diagnostics will be given:

i) Data Base Management System Errors

These errors are encountered when there may be some danger of destroying the contents of the URA data base or there is an error in the URA software. Even though the URA software might be the cause of the error, it is doubtful if it will do anything to harm the contents of the user's data base. A complete list of these error messages is given in "A Data Base Management System for URA Based on DBTG 71."

*** ERROR 16 - DATA BASE FILE INCONSISTENT

This error message is given if the user attempts to modify or retrieve information from a URA data base which has had its

---

[1] The exact manner in which this is done is installation-dependent and is given in Part II, Section 8.
[2] Part III

contents altered so that it is unusable by the URA software.

*** ERROR # n FOUND IN ROUTINE # m

An error message of this format usually designates an error in the URA software, where n is the error number. To find out which routine the error occurred in, refer to "A Data Base Management System for URA Based on DBTG 71." If the values of the variables n and m are 16 and 30 respectively, the error designates a data base inconsistency which is usually a user error. Any other errors of this form with different values should be brought to the attention of those persons maintaining the URA software.

ii) URA Command Errors

These errors are encountered in the processing of URA commands and are user errors. These diagnostics are generated when the user presents ambigious or incorrect information to the URA commands. In most cases, URA will take no action to fulfill the user's request if an error is encountered. The command must be restated in corrected form, before action is taken. All these errors are presented in the following format:

URAnnn:subroutine: error-message

where "nnn" designates the URA error number, "subroutine" denotes the subroutine in the URA software where the error occurred, and "error-message" corresponds to some diagnostics which describe the error condition.

iii) URA Input Errors

These errors are encountered when the INPUT-PSL or DELETE-PSL are used incorrectly. URA always attempts to recover from these errors unless an excessive number of errors have been encountered. Each of these errors is assigned a level number, 1 through 4. The user is allowed to make up to 24 level 1 and 24 level 2 errors, but a single level 3 or level 4 error will terminate processing of the command. The levels are described below.

Level	Description	Limit
1	This statement is ignored.	99
2	Serious user error	24
3	URA unable to recover	0
4	Exceeded URA capabilities	0

These types of errors are presented in the following format:

**** LEVEL j, URAnnn:subroutine: error-message

where "j" designates the level number and "nnn" denotes the URA error number. The last part of the format is the same as the URA command errors.

After processing any URA command, a STOP status message is given. This message designates that processing of the command was successful (STOP 0, i.e., errors were handled effectively, etc.) or that processing was not totally successful (STOP 4 or STOP 8). STOP 4 is given when an error level limit is exceeded for INPUT-PSL errors, for example. The STOP 8 message designates a serious error in attempting to access the data base, usually resulting from an inconsistent data base.

The following is a list of all possible errors that can be encountered while using URA. A short description of each error accompanies it as well as suggested action to take should the error occur. More than one error may have a similar explanation, while the software makes a distinction because the error is recognized in different places.

Number	Subroutine	Error Message
2	NLEX	NAME TOO LONG A user defined name has exceeded the 30 character limit allowed by URL/URA. The name is truncated to 30, but is still put in the data base. (See section 10.1, part v.)
3	NLEX	'EOF' NOT FOUND BEFORE END-OF-FILE The user has terminated the input before specifying the URL 'EOF'. Processing of the input is terminated.
4	INDSS	ERROR OPENING DATA BASE FOR -
5	NLEX	END-OF-FILE IN MIDDLE OF COMMENT The end of input has been encountered following the '/*' comment characters. Processing of the input is terminated.
6	SCAN	INVALID LEXICAL TYPE RETURNED FROM NLEX URA software error. Please notify persons maintaining URA should this error occur.
7	SCAN	ILLEGAL CHARACTER - IGNORED An illegal character encountered when scanning an input line. See the "User Requirements Language, Version 4.0 Language Reference Manual" for a complete list of legal characters. The illegal character is ignored and processing of the URL statement continues.
8	COMLOP	PARSE STACK OVERFLOW URA software error. Please notify persons maintaining URA should this error occur.
9	PROK	BAD CASE URA software error. Please notify persons maintaining URA should this error occur.
10	REDUCE	NO APPLICABLE PRODUCTION - SYNTAX ERROR - START SKIPPING Illegal URL statement syntax is encountered. If this is a header statement, following statements will be assigned to the previous header statement. The error may be a result of incorrect usage of a URL reserved word. (See section 10.1, part i.)

11	STACK	ILLEGAL SYMBOL PAIR - SYNTAX ERROR - START SKIPPING Illegal URL statement syntax is encountered. If this is a header statement, following statements will be assigned to the previous header statement. This statement is not entered into the URA data base. (See section 10.1, part i.)
12	ISYMBL	SYMBOL TABLE OVERFLOW Exceeded limits of URA. Reissue INPUT-PSL command at point in the input file where this error occurred.
13	ISYMBL	TOO MANY SYMBOLS Exceeded limits of URA. Reissue INPUT-PSL command at point in the input file where this error occurred.
14	SETYPE	INVALID SYMBOL TABLE POINTER URA software error. Please notify persons maintaining URA should this error occur.
15	STACK	INVALID CASE URA software error. Please notify persons maintaining URA should this error occur.
16	COMENT	END-OF-FILE IN COMMENT ENTRY End of input encountered in URL comment entry. Processing of the input is terminated.
17	SKIP	END OF FILE WHILE SKIPPING Serious error. In attempt to recover from previous errors the end of input has been encountered. Processing of input is terminated.
18	FETCH	PROCESS HAS NO RESOURCE USAGE -
19	RECOV	UNABLE TO RECOVER AT THIS TIME Processing of input is terminated due to serious errors which make it unable to continue.
20	RECOV	LAST STATEMENT SKIPPED Statement where error occurred is skipped so that processing of input may continue.
21	SETINP	INVALID SYMBOL TABLE POINTER URA software error. Please notify

persons maintaining URA should this error occur.

22 RWLIS2 SAME ATTRIBUTE ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE
An attempt was made to assign a second value to the same ATTRIBUTE for a given name. The new value is ignored.

23 UPDTRS OVERFLOW IN RESOURCE STACK -- CONTINUING

24 MAINRCOM MISSING SEMICOLON ON LINE AFTER NAME
Semicolon is needed to terminate comment entry statement.

25 HEAD INVALID HEADER STATEMENT - STATEMENTS WILL BE IGNORED
Illegal syntax of header statement. All URL statements up to the next header statement will be ignored. (This error is also described in section 10.1, part iii.)

26 IGTYPE INVALID SYMBOL TABLE POINTER
URA software error. Please notify persons maintaining URA should this error occur.

27 PTABIN INVALID LEXICAL TYPE OR END-OF-FILE
URA software error. Please notify persons maintaining URA should this error occur.

28 PTABIN DUPLICATE RESERVED WORD - IGNORED
URA software error. Please notify persons maintaining URA should this error occur.

29 MAINCT CONFLICT WITH EXISTING DATA BASE CONNECTIONS
Attempt made to change name type to one which conflicts with the context in which the name is used. No change is made.

30 UPDTRS NO MEASURES INFORMATION FOR RESOURCE -

31 MAINCT BAD INPUT FORMAT
The format of the file used as input to the command is incorrect. See the command description for correct format. No change is made.

32 MAINCT NAME NOT IN DATA BASE
Attempt to change name type of name not

defined in the URA data base.

33	MAINCT	INVALID NAME TYPE Attempt to assign an illegal name type to a name. Probably a spelling error.
34	MAINCT	NAME TYPE TOO LONG Attempt to assign an illegal name type to a name. Probably a spelling error.
35	MAINCT	WARNING - STUFF AFTER NAME TYPE The input file contains more than just name and new name type. The extra data will be ignored by the command processor.
36	MAINCT	INVALID NAME - TOO LONG The name for which the change is to be made is over 30 characters. Check spelling.
37	UPDTRS	OVERFLOW IN INFORMATION STACK - CONTINUING
38	MAINREN	OLD NAME NOT IN DATA BASE Attempt to change name of some object which is not defined in the data base. Probably a spelling error.
39	MAINREN	NEW NAME ALREADY IN DATA BASE Attempt to change old name to a name already defined in the data base. User must choose another name.
40	CLREN	MUST GIVE OLD AND NEW, OR INPUT Parameters given for the command do not supply sufficient information for processing. Reissue command.
41	MAINDEL	NAME TO BE DELETED NOT IN DATA BASE Attempt to delete a name which is not defined in the URA data base.
42	MAINDEL	INVALID MEMBER TYPE URA software error. Please notify persons maintaining URA should this error occur.
43	OTHERS	CARDINALITY ALREADY GIVEN AS SYSPAR Attempt to assign a numerical value to a CARDINALITY statement when previously assigned a SYSTEM-PARAMETER name. The value is ignored.
44	OTHERS	CARDINALITY ALREADY GIVEN AS DIFFERENT

		VALUE Attempt to assign a second value to a CARDINALITY statement. The new value is ignored.
45	CLST	NO TYPE GIVEN WITH "NAME=" OR "FILE" PARAMETER No new name type has been specified. The command must be reissued.
46	CT	TYPE, BUT NO NAME OR FILE GIVEN No name has been specified to have its name type changed. The NAME or FILE parameter must be given.
47	PRERCA	INTERVAL NOT FOUND IN DATA BASE -
48	PRERCA	KEYWORD NOT FOUND IN DATA BASE -
49	ENUMRT	NO NAMES IN DATA BASE Attempt to retrieve names from an empty data base.
50	PLIST	TOO MANY NAMES - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
51	RWLIST	MUST BE SUBSETTING CRITERION NAME Attempt to define a name which is not a GROUP or ELEMENT to be SUBSETTING CRITERION.
52	IDENTC	NAME NOT IN DATA BASE Attempt to retrieve information about a name which is not defined in the data base.
53	OPTRW	NAME LIST TOO LONG - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
54	UPDTPR	PROCESS HAS NO PROCESSOR -
55	UPDTPR	OVERFLOW IN PROCESSOR STACK - MUST ABORT
56	UPDTPR	NO HAPPENS INFORMATION FOR -
57	FETCH	SYSTEM PARAMETER HAS NO VALUE -
58	ENDRUP	NO PROCESSOR INFORMATION FOR -
59	PREPRO	ROOT PROCESS NOT FOUND IN DATA BASE -
60	APPLES	SECOND MAILBOX FOR PD ILLEGAL

		Attempt to associate a second MAILBOX to a particular PROBLEM DEFINER.
61	RWLIST	ALREADY PART OF SOMETHING ELSE Attempt to define a structure where an object is PART OF more than one other object. This is contrary to rules specified in the "User Requirements Language, Version 3.2, Language Reference Manual."[1]
62	RWLIST	SECOND PD FOR THIS ITEM ILLEGAL Attempt to assign a second RESPONSIBLE-PROBLEM-DEFINER to an object. This statement is ignored.
63	RWLIST	ALREADY PART OF SOMETHING ELSE Attempt to define a structure where an object is PART OF more than one other object. This is contrary to rules specified in the "User Requirements Language, Version 3.2, Language Reference Manual."[1]
64	MAINDICT	NAME NOT FOUND IN DATA BASE Attempt to retrieve information about a name that is not defined in the data base.
65	MAINCONT	NAME NOT FOUND IN DATA BASE - Attempt to retrieve information about a name that is not defined in the data base.
66	MAINPIC	NAME NOT IN DATA BASE Attempt to retrieve information about a name that is not defined in the data base.
67	MAINPIC	PICTURE NOT AVAILABLE FOR Attempt to generate the report for a name which is not a SET, INPUT, OUTPUT, ENTITY, GROUP, ELEMENT, PROCESS or INTERFACE. Only these types of objects may have PICTURES generated for them.
68	REPSET	WARNING - MISSING SEMICOLON. NEW COMMENT ENTRY ADDED Semicolon not given to terminate comment entry. One is assumed and processing

---

[1] Part II of the URL User's Manual.

continues.

69	REPSFT	NO NEW COMMENT ENTRY - OLD ENTRY HAS BEEN DELETED Since no new comment entry has been given to replace the old, the old comment entry statement is deleted.
70	CLDCOM	NO NAME OR FILE SPECIFIED Either the NAME or FILE parameter must be given for this command to be executed. Parameters given do not supply sufficient information for processing. Reissue command.
71	CLCT	PARAMETER LEGAL ONLY IN MTS VERSION -
72	CLCONT	PARAMETER LEGAL ONLY IN MTS VERSION -
73	CLNG	PARAMETER LEGAL ONLY IN MTS VERSION -
74	CLIP	PARAMETER LEGAL ONLY IN MTS VERSION -
75	CLFPS	PARAMETER LEGAL ONLY IN MTS VERSION -
76	MAINPRIO	NAME NOT IN DATA BASE Attempt to retrieve information about a name that is not defined in the data base.
77	CLDCOM	PARAMETER LEGAL ONLY IN MTS VERSION -
78	CLDEL	PARAMETER LEGAL ONLY IN MTS VERSION -
79	CLDICT	PARAMETER LEGAL ONLY IN MTS VERSION -
80	CLKWIC	PARAMETER LEGAL ONLY IN MTS VERSION -
81	CLPAV	PARAMETER LEGAL ONLY IN MTS VERSION -
82	CLPCOM	PARAMETER LEGAL ONLY IN MTS VERSION -
83	CLPIC	PARAMETER LEGAL ONLY IN MTS VERSION -
84	CLRCOM	PARAMETER LEGAL ONLY IN MTS VERSION -
85	CLREN	PARAMETER LEGAL ONLY IN MTS VERSION -
86	CLSTR	PARAMETER LEGAL ONLY IN MTS VERSION -
87	CLDEL	NO NAME OR FILE WAS SPECIFIED Either the NAME or FILE parameter must be given for this command to be executed. Parameters given do not supply sufficient

		information for processing. Reissue command.
88	MAINPRIO	NAME NOT A PROCESS NAME Attempt to retrieve information from a name that is not a PROCESS name. Only PROCESS names may be used as input to this command.
89	PROBE	LOOP IN SUBPART/UTILIZES STRUCTURE AT -
90	RWLIST	SSCN IS ONLY LEGAL TYPE IN DEFINE SECTION WHICH CAN BE MAINTAINED Attempt to use MAINTAINED statement for some object which is not SUBSETTING-CRITERION.
91	ADDUSE	TOO MANY USAGES URA software error. Please notify persons maintaining URA should this error occur.
92	MAINPAV	NAME NOT IN DATA BASE Attempt to retrieve information about name which is not defined in the data base.
93	MAINPAV	NAME HAS NO USAGES AS ATTRIBUTE FOR ANYTHING Attempt to retrieve ATTRIBUTE information for a name which is not an ATTRIBUTE.
94	INPAR	COMMANDS IN INCORRECT SEQUENCE
95	CLEI	MUST GIVE EITHER ENTITY OR IDENTIFIER PARAMETER Either the ENTITY or IDENTIFIER parameter must be used in conjunction with this command for successful implementation.
96	MAINSAVE	WRITE ERROR - DATA BASE NOT SAVED
97	MAINSAVE	DATA BASE NOT SAVED - FILE CANNOT BE EMPTIED -
98	CONCOL	NAME NOT IN DATA BASE Attempt to retrieve information about a name not defined in the data base.
99	IDENTR	NAME NOT IN DATA BASE Attempt to retrieve information about a name not defined in the data base.
100	NLIST2	TOO MANY ATTRIBUTE VALUE PAIRS IN SINGLE

		STATEMENT Limit exceeded. Remaining pairs should be given in another statement.
101	NLIST2	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name defined as something else as an ATTRIBUTE name.
102	NLIST2	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name defined as something else as an ATTRIBUTE-VALUE name.
103	DELSET	DESCRIPTION COMMENT ENTRY NOT FOUND FOR: Attempt to delete a nonexistent DESCRIPTION statement.
104	DELSET	PROCEDURE COMMENT ENTRY NOT FOUND FOR: Attempt to delete a nonexistent PROCEDURE statement.
105	DELSET	VOLATILITY COMMENT NOT FOUND FOR: Attempt to delete a nonexistent VOLATILITY statement.
106	DELSET	VOLATILITY-MEMBER COMMENT ENTRY NOT FOUND FOR: Attempt to delete a nonexistent VOLATILITY-MEMBER statement.
107	DELSET	VOLATILITY-SET COMMENT ENTRY NOT FOUND FOR: Attempt to delete a nonexistent VOLATILITY-SET statement.
108	DELSET	DERIVATION COMMENT ENTRY NOT FOUND FOR: Attempt to delete a nonexistent DERIVATION statement.
109	DELSET	TRUE WHILE COMMENT ENTRY NOT FOUND FOR: Attempt to delete a nonexistent TRUE WHILE statement.
110	DELSET	FALSE WHILE COMMENT NOT FOUND FOR: Attempt to delete a nonexistent FALSE WHILE statement.
111	MAINDCOM	NAME NOT FOUND IN DATA BASE Attempt to delete information for a name not defined in the data base.
112	PUNSET	NO DESCRIPTION AVAILABLE FOR :
113	CLCM	MUST GIVE EITHER CONSISTS OR CONTAINED

		PARAMETER Either the CONSISTS or CONTAINED parameter must be used in conjunction with this command.
114	VLIST	ONLY SINGLE VALUE OR RANGE ALLOWED - IGNORED Invalid format for specifying a VALUES statement. See the "User Requirements Language, Version 3.2, Language Reference Manual."[1]
115	VLIST	MIN NOT LESS THAN MAX - IGNORED If a number range is specified the first number must be less than the second number.
116	OTHERS	VALUES ONLY LEGAL FOR ELEMENT, SYSPAR, or ATTRIBUTE-VALUE Attempt to use a VALUES statement for a name which is not an ELEMENT, SYSTEM-PARAMETER or ATTRIBUTE-VALUE.
117	OTHERS	DIFFERENT VALUES ALREADY GIVEN Attempt to assign a second value for a given object. This statement is ignored.
118	CLRCA	FILE= NOT ALLOWED IN THIS IMPLEMENTATION Error encountered in using a SYSTEM-PARAMETER in a given statement. Interpretation of rest of statement becomes confused.
119	CLRCA	PUNCH= NOT ALLOWED IN THIS IMPLEMENTATION Attempt to use zero as a SYSTEM-PARAMETER.
120	SNAMET	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name in a context which conflicts in the way it has previously been used.
121	MAINRCOM	NAME NOT FOUND IN DATA BASE Attempt to access information for a name not defined in the data base.
122	MAINRCOM	INVALID TYPE OF COMMENT ENTRY Attempt to replace unrecognizable comment entry statement. Probably a spelling error.

---

[1] Part II of the URL User's Manual.

123	MAINRCOM	CANNOT HAVE THIS TYPE OF COMMENT ENTRY Attempt to assign a comment entry statement which is not legal for the particular name type.
124	REPSET	...WITH THIS NAME Used in conjunction with URA123. Specifies the name for which the comment entry was used.
125	MAINRCOM	PROBLEMS SCANNING INPUT FILE - MUST ABORT Incorrect format of file used for input. See command description for correct format.
126	REPSET	WARNING - THERE IS NO COMMENT ENTRY TO DELETE Attempt to delete nonexistent comment entry.
127	PUNSET	DESCRIPTION COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent DESCRIPTION statement.
128	PUNSET	PROCEDURE COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent PROCEDURE statement.
129	PUNSET	VOLATILITY COMMENT NOT FOUND FOR: Attempt to retrieve nonexistent VOLATILITY statement.
130	PUNSET	VOLATILITY-MEMBER COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent VOLATILITY-MEMBER statement.
131	PUNSET	VOLATILITY-SET COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent VOLATILITY-SET statement.
132	PUNSET	DERIVATION COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent DERIVATION statement.
133	PUNSET	TRUE WHILE COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent TRUE WHILE statement.
134	PUNSET	FALSE WHILE COMMENT NOT FOUND FOR: Attempt to retrieve nonexistent FALSE WHILE statement.

135	MAINPCOM	NAME NOT FOUND IN DATA BASE Attempt to retrieve information for a name not defined in the data base.
136	PRLMT	NO ELEMENTS SATISFY THE REQUESTED ATTRIBUTE-VALUES
137	MAINVAL	NAME NOT IN DATA BASE -
138	MAINVAL	NAME IS NOT AN ATTRIBUTE OR IT HAS NO VALUE -
139	MAINVAL	NO NAMES IN DATA BASE
140	MANDP	VALUE OF MM IS GREATER THAN THE NUMBER OF MAND. ATTR. - IGNORED
141	CONROW	NAME NOT IN DATA BASE Attempt to retrieve information for a name not defined in the data base.
142	CLRCA	EMPTY NOT ALLOWED IN THIS IMPLEMENTATION Attempt to delete a relationship, between two names, which is not defined in the data base.
143	MAINDP	NAME NOT IN DATA BASE Attempt to retrieve information about a name which is not defined in the data base.
144	DPCOL	TOO MANY COLUMNS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
145	DPCOL	TOO MANY ROWS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
146	DPCOL	SPARSE MATRIX SYSTEM OVERFLOW Exceeded limits of the software that produces the matrix.
147	DPROW	TOO MANY ROWS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
148	DPROW	TOO MANY COLUMNS

Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.

149 DPPOW SPARSE MATRIX SYSTEM OVERFLOW
Exceeded limits of the software that produces the matrix.

150 DPSUM NO ROWS
No relationships can be specified about the names used as input, so no matrix will be generated.

151 DPSUM NO COLUMNS
No relationships can be specified about the names used as input, so no matrix will be generated.

152 DPSUM SPARSE MATRIX SYSTEM OVERFLOW
Exceeded limits of the software that produces the matrix.

153 MAINDP INVALID INPUT NAME TYPE
Attempt to use a name which is not a PROCESS name as input to the command.

154 MAINDP INVALID INPUT NAME TYPE
Attempt to use a name which is not a SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT name as input to the command.

155 DPSUM INVALID ROW TYPE - SYSTEM ERROR
URA software error. Please notify persons maintaining URA should this error occur.

156 CLRCA NO INTERVAL GIVEN - MUST ABORT
Attempt to delete a relationship which has not been defined in the data base.

157 CLRCA NBP AND NBR NOT ALLOWED TOGETHER - MUST ABORT
Attempt to delete a relationship which has not been defined in the data base.

158 DBTCON NAME NOT IN DATA BASE
Attempt to delete a relationship which has not been defined in the data base.

159 DBTCON NAMES NOT CONNECTED IN THAT FASHION
Attempt to delete a relationship which has not been defined in the data base.

160	UDDERS	NAME NOT IN DATA BASE Attempt to delete a relationship which has not been defined in the data base.
161	UDDERS	NO CONNECTIVITY EXISTS Attempt to delete a relationship which does not exist for this name.
162	UDDERS	DIFFERENT CONNECTIVITY IN DATA BASE - NOT DELETED Attempt to delete a relationship which is not stated exactly as it is in the data base.
163	PRTPR	NO PROCESSOR INFORMATION FOUND IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
164	PRTRS	NO PROCESSOR INFORMATION FOUND IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
165	UPDTPR	SYSTEM PARAMETER WITHOUT VALUE - Attempt to delete a relationship which is not defined in the data base.
166	UDDERS	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
167	UDDERS	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
168	UDDERS	DIFFERENT VALUES IN DATA BASE- NOT DELETED Attempt to delete a number or range of numbers that was not defined for the statement.
169	ADJINT	HAPPENS INFORMATION NOT CONVERTIBLE FOR THE INTVL - Attempt to delete a relationship which is not defined in the data base.
170	ADJINT	SYSTEM PARAMETER WITHOUT VALUE ENCOUNTERED - Attempt to delete a relationship which is not defined in the data base.
171	DELSYN	NAME NOT IN DATA BASE Attempt to delete a relationship which is

not defined in the data base.

172	DELSYN	NAME IS NOT A SYNONYM FOR THIS NAME Attempt to delete a SYNONYM relationship which is not defined in the data base.
173	DRIVFS	USES INFORMATION NOT IN DATA BASE Attempt to delete a relationship which is not specified exactly as it is in the data base. Relationship is not deleted.
174	DRIVES	WARNING - "USING" INFO IN A DATA BASE Statement deleted although the relationship has not been specified exactly as in the data base (the "USING" clause has been omitted).
175	DRIVES	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
176	DRIVES	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
177	DRIVES	THESE TWO NAMES NOT CONNECTED IN THAT WAY Attempt to delete a relationship which is not defined in the data base.
178	DHAPNS	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
179	DRIVFS	"USING" NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
180	DHAPNS	NAMES NOT CONNECTED IN THAT FASHION Attempt to delete a relationship which is not defined in the data base.
181	DISCON	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
182	DISCON	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
183	DISCON	NAMES NOT CONNECTED IN THAT FASHION Attempt to delete a relationship which is not defined in the data base.
184	ATTV	NAME DOESN'T HAVE THIS ATTRIBUTE

		Attempt to delete a relationship which is not defined in the data base.
185	ATTV	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
186	ATTV	NAME HAS NO ATTRIBUTES Attempt to delete ATTRIBUTE relationship for a name with no ATTRIBUTES.
187	DCNSIS	SYSPAR VALUE IN DATA BASE IS DIFFERENT - IGNORED Attempt to delete a statement using a different SYSTEM-PARAMETER. Statement not deleted.
188	DCNSIS	WARNING - SYSPAR IN DATA BASE Statement deleted though user did not include SYSTEM-PARAMETER with statement.
189	DCNSIS	NAME NOT IN DATA BASE Attempt to delete a CONSISTS relationship using a name not defined in the data base.
190	DCNSIS	CONSISTS/CONTAINED INFORMATION NOT IN DATA BASE Attempt to delete a CONSISTS or CONTAINED relationship not defined in the data base.
191	DCNSIS	NO SYSPAR IN DATA BASE - IGNORED Attempt to delete a relationship which is not defined exactly in the same way as defined in the data base. Statement not deleted.
192	PRERCA	NAME GIVEN AS KEYWORD IS NOT A KEYWORD - Attempt to delete a relationship which is not defined in the data base.
193	PRERCA	NAME GIVEN AS AN INTVL IS NOT A INTVL -
194	PREPRO	NAME GIVEN AS PROCESS IS NOT A PROCESS -
195	MAINSECA	NAME NOT IN DATA BASE - Attempt to delete a relationship which is not defined in the data base.
196	PLONG	COMMENT NOT FOUND ***SYSTEM ERROR *** URL software error. PLease notify persons maintaining URL should this error occur.

197	COMNT	COMMENT-ENTRIES NOT ALLOWED IN DPSL Attempt to delete comment-entry statements. This can only be done using the DCOM and RCOM commands. Statement not deleted.
198	COMNT	EOF WHILE LOOKING FOR SEMICOLON Improper statement syntax has been encountered. Semicolons are needed to end all URL statements.
199	DHAPNS	DIFFERENT SYSPAR - NOT DELETED Attempt to delete a HAPPENS relationship using a different SYSTEM-PARAMETER than defined in the data base. Statement not deleted.
200	DHAPNS	INTERVAL NOT IN DATA BASE Attempt to delete a HAPPENS relationship using an INTERVAL not defined in the data base.
201	PLIST	NAME NOT PART OF HEADER An illegal statement header has been given. Probably a spelling error. The statement is ignored.
202	NLIST	NAME PREVIOUSLY USED DIFFERENTLY - IGNORED Attempt to use a name in a context different than the way it is defined. (This error is also described in section 10.1.)
203	MAINSECA	INPUT NAME HAS INVALID TYPE Attempt to delete a relationship not defined in the data base.
204	DEFN	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to assign a name type to a name which is used in a different context.
205	SETSYN	ALREADY SYNONYM FOR SOMETHING ELSE Attempt to assign a name to be a SYNONYM for more than one object.
206	SETSYN	UNABLE TO MAKE SYNONYM - TOO COMPLICATED See section 10.1, part vii for explanation and solution to this error.
207	SETSYN	CANNOT BE MADE SYNONYM - DIFFERENT TYPES Attempt to assign a name as a SYNONYM to a different name, both with different

name types.

208	MAINNLA	NO NAMES IN DATA BASE
209	CHKCON	STACK OVERFLOW WHILE WALKING CONSISTS STRUCTURE Attempt to use a name which is not an INTERVAL in a CONSISTS statement for an INTERVAL section.
210	PRTNUN	NO NAMES IN DATA BASE Attempt to use a name in a context different than the way the name is defined.
211	OTHERS	NAME MUST BE ENTITY NAME Attempt to use a name in a context where only an ENTITY name is acceptable.
212	OTHERS	RELATION ALREADY EXISTS BETWEEN TWO OTHER ENTITIES Attempt to specify the same RELATION for a different pair of ENTITIES. Different ENTITY pairs imply different RELATIONS.
213	OTHERS	CAN ONLY HAVE ONE CARDINALITY Attempt to specify a second CARDINALITY statement for a name. Objects may have only one CARDINALITY.
214	OTHERS	CONNECTIVITY ALREADY GIVEN FOR THIS RELATION Attempt to specify a second CONNECTIVITY statement for a name. RELATIONS may have only one CONNECTIVITY.
215	RWLIS2	ALREADY CONTAINS WITH DIFFERENT SYSTEM PARAMETER Attempt to specify the same CONSISTS statement, but with two different SYSTEM-PARAMETERS.
216	OTHERS	NAME MUST BE ENTITY NAME BEFORE VIA Attempt to use a name in a statement where only an ENTITY name is allowed.
217	OTHERS	NAME MUST BE RELATION AFTER VIA Attempt to use a name in a statement where only a RELATION name is allowed.
218	OTHERS	RELATION ALREADY EXISTS BETWEEN DIFFERENT ENTITY PAIR Attempt to specify the same RELATION for a different pair of ENTITIES. Different

ENTITY pairs imply different RELATIONS.

219 CLSECA FILE= NOT ALLOWED IN THIS IMPLEMENTATION
Attempt to use a name in a statement where only a CONDITION name is allowed.

220 SETSYN CANNOT MAKE A NAME FOR ITSELF
Attempt to specify a basic name as a synonym for itself. Basic names cannot also be synonyms.

221 NLIST TOO MANY NAMES - REST IGNORED
Attempt to specify a list of names for a statement where only a single name is acceptable.

225 RWLIST CANNOT HAVE KEYWORD FOR KEYWORD
Attempt to assign a KEYWORD to a KEYWORD name.

228 RWLIST CANNOT HAVE SECURITY FOR SECURITY
Attempt to assign a SECURITY statement to a SECURITY name.

229 RWLIST CANNOT HAVE SOURCE FOR SOURCE
Attempt to assign a SOURCE to a SOURCE name.

231 RWLIST SYNONYMS ONLY APPLIED TO FIRST NAME
Attempt to assign a SYNONYM to more than one name. The SYNONYM is given only to the first name.

232 APPLES APPLIES STATEMENT ILLEGAL WITH THIS NAME TYPE
Attempt to use APPLIES statement for a name which is not a KEYWORD, MAILBOX, SECURITY or SOURCE.

233 DEFN TOO MANY NAMES IN DEFINE HEADER - REST IGNORED
Exceeded 50 name limit, remaining names should be given in another statement.

234 OPTRW NAME ALREADY USED IN DIFFERENT CONTEXT
Attempt to use a name in a wrong context. Only a PROCESS name can be used in this context.

235 OPTION NAME ALREADY USED IN DIFFERENT CONTEXT
Attempt to use a name in wrong context for an UPDATES relationship.

236 OPTION NAME LIST TOO LONG - REST IGNORED

240	APPLES	KEYWORD CANNOT APPLY TO KEYWORD Attempt to use the APPLIES statement in the wrong context.
241	APPLES	MAILBOX CAN ONLY APPLY TO PD Attempt to use the APPLIES statement in the wrong context.
242	QROPEN	ATTEMPT TO OPEN TOO MANY Q FILES - SYSTEM ERROR
243	QWOPEN	ATTEMPT TO OPEN TOO MANY Q FILES - SYSTEM ERROR
244	INPARA	COMMANDS IN INCORRECT SEQUENCE
245	GETSEC	*** SYSTEM ERROR
246	APPLES	SECURITY CANNOT APPLY TO SECURITY Attempt to use the APPLIES statement in the wrong context.
247	APPLES	SOURCE CANNOT APPLY TO SOURCE Attempt to use the APPLIES statement in the wrong context.
248	APPLES	MEMO CANNOT APPLY TO MEMO Attempt to use the APPLIES statement in the wrong context.
249	APPLES	INVALID SECTION - WOOPS URA software error. Please notify persons maintaining URA should this error occur.
250	LIOFRE	*** SYSTEM ERROR
251	SNAMET	ATTEMPT TO CHANGE TYPE WHEN ALREADY TYPED URA software error. Please notify persons maintaining URA should this error occur.
252	SETSYN	SYNONYM TABLE OVERFLOW Exceeded URA limits. The user should reissue INPUT-PSL command at point in the input file where this error occurred.
253	RWLIST	INVALID STATEMENT NUMBER URA software error. Please notify persons maintaining URA should this error occur.
254	NAMCR	*** SYSTEM ERROR *** CANNOT CREATE

		SYNONYM URA software error. Please notify persons maintaining URA should this error occur.
255	SYNTH	NUMBER NOT ALLOWED AS SYSPAR THIS VERSION OF URA
256	RWLIST	CONNECTION ALREADY EXISTS WITH DIFFERENT VALUE OR NAME URA software error. Please notify persons maintaining URA should this error occur.
257	FORMSL	NAME NOT IN DATA BASE -
260	SORTMD	*** SYSTEM ERROR
265	RWLIST	CONNECTION ALREADY EXISTS WITH DIFFERENT VALUE OR NAME Attempt to specify same relationship between two INTERVAL names though with different SYSTEM-PARAMETER. Not allowed.
266	ILLST	ILLEGAL STATEMENT IN THIS SECTION Attempt to use a URL statement in a wrong context. See "User Requirements Language, Version 4.0, Language Reference Manual".[1]
267	ILLST	NO CURRENT SECTION Attempt to use an illegal section header statement. See "User Requirements Language, Version 4.0, Language Reference Manual".[1]
269	NLIST	NAME LIST TOO LONG, REST IGNORED Limit of 50 names has been exceeded. Remaining names should be given in another statement.
270	INPAR	ERROR OPENING DATA BASE - MUST ABORT Attempt to use an inconsistent data base. No processing can be done on it. (See section 10.1.)
271	MAINCNC	NAME NOT IN DATA BASE Attempt to retrieve information for a name not defined in the data base.
272	CNCBLD	TOO MANY ROWS - STOPPING HERE

---

[1] Part II of the URL User's manual.

		Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CNC command.
273	CNCBLD	NAME DOESN'T CONSIST OF ANYTHING No information can be presented for this name in the matrix.
274	CNCBLD	TOO MANY LEVELS - LOWER LEVEL STUFF IGNORED Too many levels of CONSISTS information to be presented.
275	CNCBLD	***THE FOLLOWING NAMES ARE INVOLVED IN A LOOP: This problem should be corrected by modifying the CONSISTS statements for these names.
276	CNCBLD	TOO MANY LEVELS - LOWER LEVEL STUFF IGNORED Too many levels of CONSISTS information to be presented.
277	CNCBLD	TOO MANY COLUMNS - STOPPING HERE Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-COMPARISON command.
278	CNCBLD	SPARSE MATRIX OVERFLOW - STOPPING HERE Exceeded limits of software that produces the matrix.
279	CNCSUM	***NO COLUMNS, OR NO ROWS - STOPPING No relationships can be specified about the names used as input, so no matrix will be generated.
280	CNCSUM	LESS THAN 2 ROWS, NO SIMILARITY MATRIX Not enough information is available to generate a matrix.
281	CNCSUM	SPARSE MATRIX OVERFLOW - STOPPING Exceeded limits of software that produces the matrix.
282	MUST	STACK OVERFLOW - CONTINUING Exceeded limits of software that produces the report. An attempt is made to recover and process as much data as possible.

283	HAVE	STACK OVERFLOW - CONTINUING Exceeded limits of software that produces the report. An attempt is made to recover and process as much data as possible.
284	MAINSTR	TOO MANY LEVELS - CONTINUING Exceeded limits of software that produces the report. An attempt is made to process as much data as possible.
285	PRRPS	THE FOLLOWING NAMES ARE INVOLVED IN LOOPS Through incorrect specification of PARTS/SUBPARTS statements, loops have been implied in structures of objects in the problem statement. The user should determine which PARTS/SUBPARTS relationships should be changed and delete them via the DELETE-PSL command.
286	PRINTV	NO FREQUENCY INFORMATION IN DATA BASE No HAPPENS statements have been specified in the problem statement stored in the data base. If any output is desired from this report, at least one HAPPENS statement must be in the data base.
287	MAINIDX	NO NAMES IN INDEX Attempt to generate an index into a report presenting no information about any names. This is merely a warning.
288	STATPS	NO NAMES AT LEVEL ONE Attempt to generate STRUCTURE report for names of particular name type (i.e., PROCESS, INPUT, OUTPUT or INTERFACE), but no names of this type currently exist in the data base. To generate this report for PROCESS names, for example, at least one PROCESS must be defined in the data base.
289	PCLRBT	NO PICTURE AVAILABLE FOR Attempt to generate a PICTURE for names which legally have a PICTURE, but no information that was specified for this name can be presented in PICTURE format. Information that may be presented in a PICTURE is any dealing with interaction of data and PROCESSES and structure (CONSISTS and SUBPARTS statements).
290	SETSYN	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to assign a name of UNDEFINED

		name type as a SYNONYM to another name which has been used in some context in conflict with the manner in which the UNDEFINED name has been used.
291	INPDUM	ERROR OPENING DATA BASE FOR -
292	INPDUM	DATA BASE IS EMPTY
293	MAINPRES	INVALID INPUT CODE
294	INPRES	ERROR OPENING DATA BASE FOR -
295	INPRES	DATA BASE MUST BE EMPTY
296	ABPRES	****** RUN ABORTED ******
297	INPRES	NULL OR INVALID FIRST CARD
298	GETDMC	SEQUENCE ID OUT OF ORDER
299	GETDMP	SEQUENCE ID OUT OF ORDER
300	PUTRNG	INVALID RANGE SPEC
301	IDENTR	***TOO MANY COLUMNS -- MUST STOP HERE*** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.
302	IDENTR	***TOO MANY ROWS -- MUST STOP HERE*** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.
303	IDENTR	***MATRIX OVERFLOW -- MUST STOP HERE*** Exceeded limits of software that produces the matrix.
304	IDENTR	THE FOLLOWING NAMES DO NOT IDENTIFY ANYTHING: No information can be presented in the matrix for these names because they do not "IDENTIFY" any ENTITIES.
305	IDENTC	***TOO MANY COLUMNS -- MUST STOP HERE*** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.
306	IDENTC	***TOO MANY ROWS -- MUST STOP HERE***

AD-A057 130 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS ANALYZER (URA) USER'S MANUAL H6180/MULTICS/VE--ETC(U)
MAR 77 D TEICHROEW, A HERSHEY, S SPEWAK F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-128 NL

2 OF 6
AD A057130



Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.

307 IDENTC ***MATRIX OVERFLOW -- MUST STOP HERE***
Exceeded limits of software that produces the matrix.

308 IDENTC THE FOLLOWING NAMES ARE NOT IDENTIFIED BY ANYTHING:
No information can be presented in the matrix for these names because no "IDENTIFIED" relationships have been specified for them.

309 CONROW ***TOO MANY COLUMNS -- MUST STOP HERE***
Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.

310 CONROW ***TOO MANY ROWS -- MUST STOP HERE***
Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.

311 CONROW THE FOLLOWING ARE NOT CONTAINED IN ANYTHING:
No information can be presented in the matrix for these names because no "CONTAINED IN" relationships have been specified for them.

312 CONROW ***MATRIX OVERFLOW -- MUST STOP HERE***
Exceeded limits of software that produces this matrix.

313 CONCOL ***TOO MANY COLUMNS -- MUST STOP HERE***
Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.

314 CONCOL ***TOO MANY ROWS -- MUST STOP HERE***
Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.

315 CONCOL THE FOLLOWING DO NOT CONSIST OF ANYTHING:
No CONSISTS statements have been used in conjunction with the names listed.

316	CONCOL	***MATRIX OVERFLOW -- MUST STOP HERE*** Exceeded limits of software that produces this matrix.
317	CHKRPL	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name in a context different from its initial context.
318	RCOD5	INVALID T CODE - SYSTEM ERROR
319	RCOD5	INVALID T CODE - SYSTEM ERROR
320	RCOD5	INVALID T CODE - SYSTEM ERROR
321	RCOD6	INVALID T CODE - SYSTEM ERROR
322	RCOD6	INVALID T CODE - SYSTEM ERROR
323	RCOD6	INVALID T CODE - SYSTEM ERROR
324	RCOD6	MISSING CODE 10 CARD - SYSTEM ERROR
325	CLCM	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
326	CLCNC	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
327	CLCONT	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
328	CLCT	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
329	CLDCOM	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
330	CLDEL	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
331	CLDICT	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
332	CLDP	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
333	CLDP	MUST GIVE DATA OR PROCESS PARAMETER
334	CLDPSL	INPUT= NOT ALLOWED IN THIS IMPLEMENTATION
335	CLEI	FILE= PARAMETER NOT ALLOWED IN THIS IMPLEMENTATION
336	CLFPS	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
337	CLIP	INPUT= NOT ALLOWED IN THIS IMPLEMENTATION
338	CLKWIC	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
339	CLNG	EMPTY NOT ALLOWED IN THIS IMPLEMENTATION

340	CLPCOM	EMPTY NOT ALLOWED IN THIS IMPLEMENTATION
341	CLPAV	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
342	CLPCOM	PUNCH= NOT ALLOWED IN THIS IMPLEMENTATION
343	CLPIC	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
344	CLPRIO	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
345	CLRCOM	INPUT= NOT ALLOWED IN THIS IMPLEMENTATION
346	CLREN	INPUT= NOT ALLOWED IN THIS IMPLEMENTATION
347	CLPPS	PUNCH= NOT ALLOWED IN THIS IMPLEMENTATION
348	CLPPS	EMPTY NOT ALLOWED IN THIS IMPELEMENTATION
349	CLNG	PUNCH= NOT ALLOWED IN THIS IMPLEMENTATION
350	CLPRIO	PUNCH= NOT ALLOWED IN THIS IMPLEMENTATION
351	CLPRIO	EMPTY NOT ALLOWED IN THIS IMPLEMENTATION
352	RCOD1	RAN OUT OF ROOM IN DATA BASE - MUST ABORT
353	RCOD3	RAN OUT OF ROOM IN DATA BASE - MUST ABORT
354	RCOD4	RAN OUT OF ROOM IN DATA BASE - MUST ABORT
355	RCOD5	RAN OUT OF ROOM IN DATA BASE - MUST ABORT
356	RCOD6	RAN OUT OF ROOM IN DATA BASE - MUST ABORT
357	CLPRIO	PUNCH AND EMPTY PARAMETERS NOT IMPLEMENTED - IGNORED
358	FORMSL	ILLEGAL NAME TYPE FOR FPS
359	GETSLV	NO NAMES IN DATA BASE
384	STNDUP	STACK OVERFLOW - SYSTEM LIMITATION
389	CLPC	HORIZONTAL BOXES TOO LARGE FOR NUMBER OF COLUMNS ON PAGE The number of horizontal boxes specified will not fit in the number of columns specified.
390	CLPC	VERTICAL BOXES TOO LARGE FOR NUMBER OF ROWS ON PAGE The number of vertical boxes specified will not fit in the number of rows per page specified.

391	FILMAT	NAME NOT IN DATA BASE -
392	FILMAT	NAME NOT EVENT OR PROCESS
393	FILMAT	SPARSE MATRIX TABLE OVERFLOW, PICTURE NOT COMPLETE The process chain has gotten too large for the software to handle. This is a system limitation, and to resolve this, the number of links must be decreased.
394	REL1	STACK OVERFLOW, PICTURE NOT COMPLETE The number of successors for the given input name is too great. This is a software limitation.
395	FILMAT	LINK LIMIT SPECIFIED WAS REACHED This is not so much an error, as a note. To obtain a more complete process chain picture, the maximum number of links must be increased.
396	CLEP	HORIZONTAL BOXES TOO LARGE FOR NUMBER OF COLUMNS ON PAGE The number of horizontal boxes specified conflicts with the number of columns specified.
397	CLEP	VERTICAL BOXES TOO LARGE FOR NUMBER OF ROWS ON PAGE The number of vertical boxes specified conflicts with the number of rows per page specified.
398	CLEP	NEITHER DATA-FLOW NOR STRUCTURE WAS SPECIFIED Either DATA-FLOW or STRUCTURE must be specified (but not both).
399	CLEP	NEITHER FORWARD NOR BACKWARD WAS SPECIFIED Either FORWARD or BACKWARD must be specified (but not both).
408	CLEP	NEITHER UPWARD NOR DOWNWARD WAS SPECIFIED Either UPWARD or DOWNWARD must be specified (but not both).
409	MAINEP	NAME NOT FOUND IN DATA BASE
410	EPSUCC	NAME NOT ACCEPTABLE TYPE FOR EP REPORT The name given does not have the correct name type for this report. The possible

		name types are: PROCESS, REAL WORLD ENTITY, INPUT, ELEMENT, GROUP, OUTPUT, ENTITY, SET, UNDEFINED.
411	EPSUCC	LINK LIMIT SPECIFIED WAS REACHED This is not so much an error, as a note. To obtain a more complete Extended Picture, the maximum number of links must be decreased.
412	EPSUCC	SPARSE MATRIX TABLE OVERFLOW, PICTURE NOT COMPLETE Due to software limitations, the Extended Picture could not be completed. To resolve this, the maximum number of links must be decreased.
413	REL2	STACK OVERFLOW, PICTURE NOT COMPLETE The number of successors for the input name given is too great. This is a software limitation.
414	REL3	STACK OVERFLOW, PICTURE NOT COMPLETE The number of successors for the input name given is too great. This is a software limitation.
415	CLEP	BOTH DATA-FLOW AND STRUCTURE SPECIFIED Either DATA-FLOW or STRUCTURE must be specified, but not both.
416	CLEP	CONFLICTING PARAMETERS, FORWARD AND BACKWARD OR UPWARD AND DOWNWARD Either FORWARD or BACKWARD must be specified, but not both. The same applies to UPWARD and DOWNWARD.
417	CLPC	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
418	CLEP	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
428	COMENT	DATA BASE OVERFLOW
429	COMRW	DATA BASE OVERFLOW
430	NAMCR	DATA BASE OVERFLOW
431	NUMCR	DATA BASE OVERFLOW
479	MAINCT	WARNING - ERRORS WERE ENCOUNTERED
481	CLILOP	INVALID COMMAND -
504	F3DCON	NAME NOT IN DATA BASE

505	P3DCON	NAMES NOT CONNECTED IN THAT FASHION
511	APPLES	TRACE-KEY CANNOT APPLY TO TRACE-KEY
512	RWLIST	CANNOT HAVE TRACE-KEY FOR TRACE-KEY
513	ASSERT	TOO MANY TRIPLES, REST IN THIS STMT IGNORED
514	RWLIS2	ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE
515	UNASRT	NAMES NOT CONNECTED
521	DOOPTR	UNBALANCED PARENTHESES IN SELECTION STRING
522	GETATR	ATTRIBUTE DOESN'T APPLY TO ANYTHING There are no names which have the given ATTRIBUTE.
523	GETNML	NO NAMES WHICH MATCH CRITERION There are no names in the data base which satisfy the selection criteria.
524	NGEVAL	*** SOFTWARE ERROR
525	NGPRS	NAME ON RIGHT DOES NOT MATCH TYPE ON LEFT
526	NGPRS	INVALID SELECTION STRING The selection string given as a parameter for the Name-Gen report is not a legal boolean expression.
527	NGPRS	INVALID ITEM IN SELECTION STRING The item given in the selection string is not a legal operand or a legal operator.
528	PPEPAR	TOO MANY ORDER PARAMETERS More than five ORDER parameters have been given, or the ORDER parameters have been given incorrectly.
529	PREPAR	NAME IN ORDER LIST NO ATTRIBUTE The name given in the order list is not an ATTRIBUTE.
530	PREPAR	NAME IN ORDER LIST NOT IN DATA BASE
531	PREPAR	NAME IN ORDER LIST TOO LONG The name given has more than 30 characters.

532	GETNMR	NO NAMES WHICH MATCH CRITERION There are no names in the data base which satisfy the selection criteria.
533	NGPRS	*** SOFTWARE ERROR
534	OPTMZ	*** SOFTWARE ERROR
535	NGPRS	TOO MANY LEVELS, MAX OF 50 ALLOWED Too many levels have been specified via the SUBPARTS-OF and SUBLEVEL paramter.
553	CONSM	NAME MUST BE RESOURCE-USAGE-PARAMETER
554	CONSM	NAME LIST TOO LONG - REST IGNORED
560	RWLIS2	INCONSISTENCY IN CLASSIFICATION STATEMENT
561	RWLIS2	INCONSISTENCY IN SECURITY-ACCESS-RIGHT STATEMENT
562	RWLIS2	INCONSISTENCY IN RESOURCE-USAGE STATEMENT
563	RWLIS2	INCONSISTENCY IN RESOURCE-USAGE-PARAMETER-VALUE STATEMENT
565	IGKEY	SOFTWARE ERROR *** SHOULD NOT OCCUR ***
569	RWLIS2	INCONSISTENCY IN CONSUMES/CONSUMED STATEMENT

10. HOW TO CORRECT ERRORS

Once error situations are detected, there must be some method to deal with them. When the errors are caused by problems in generating a report, no action need be taken as no harm will come to the data base. The Report Command can simply be restated in correct format to solve the problem. If, however, an error is encountered in making modifications to the data base (via Modifier Commands) then some immediate action should be taken if the problem definer desires to maintain a correct and complete problem statement.

The errors discovered in making modifications to the data base can be "Input Errors" which are errors discovered by URA in its attempt to process the information needed to update the data base. All these errors are specified by one or more URA error messages. The majority of these errors occur in the process of using the INPUT-PSL command.

The errors discovered in the problem statement by the problem definer are called "Logical Errors." No error diagnostics are generated by URA to denote that an error has occurred. If a name was misspelled in the input information used for INPUT-PSL, the name could be legal by URL/URA conventions yet not correct from the problem definer's standpoint. "BATCH" and "BATHC" are both names that would be perfectly acceptable to URA but not to the problem definer.

The following two sections deal with aiding the problem definer in correcting both Input Errors and Logical Errors should they occur. Treatment of the error correction methodology is still at a cursory level and no attempt is made to present procedures to correct all possible errors.

10.1 Input Errors

As stated before, all input errors cause URA error diagnostics to be printed. There are a few classes of errors which happen again and again and so will be described below.

Inconsistent Data Base

This error is usually identified by getting the URA error: "URA270: INPAR: ERROR OPENING DATA BASE." This error might occur after issuing a URA Modifier or Report Command and it specifies that the contents of the file being used as a URA data base cannot be accessed by the URA software. Methods for correcting this situation are installation-dependent since it involves the manner in which files are created and initialized. See section 10.1 of Part II for solutions to this problem at a particular installation.

URA Statement Errors

These errors account for the majority of the errors encountered when inputting information into the data base via INPUT-PSL. These errors are caused by improper use of URL statements according to the rules specified in the "User Requirements Language, Version 3.2, Language Reference Manual."[1] An occurrence of any of these errors results in the statement, where the error occurred, being ignored by the system.

The "$" character printed by URA is usually fairly close in pointing out where the error occurred. Some of the more common errors (and solutions) are presented here in hopes that the users will be able to apply the methods of solving these errors to their own, specific needs.

i) Syntax Errors

These errors are often encountered through misspellings, improper format of the statement or improper usage of URL reserved words. URA usually generates either of the two error messages:

URA010:REDUCE: NO APPLICABLE PRODUCTION-SYNTAX ERROR-START SKIPPING

or,

URA011:STACK: ILLEGAL SYMBOL PAIR-SYNTAX ERROR-START SKIPPING

For example, if upon misspelling the RECEIVES statement:

```
RECEIVES FOLDER-A,FOLDER-B;
```

URA will react by printing the URA010 error message and skip that statement to go on to the next. There are some further problems that can then occur. If the error occurred in a header statement, such as PROCESS, then the header statement is skipped and all statements intended to be related to the header statement will be related to the previous header statement. When a reserved word is misspelled, URA has no way of knowing if the statement was to be a header statement or not. Take the example:

```
GROUP: G1;
CSTS: E1, E2, G2;
PROCCESS: P1;
RCVS: I1, I2;
SUBPARTS: P2, P3;
ELEMENT: E1, E2;
```

---

[1] Part II, URL User's Manual.

PROCESS has been misspelled which results in having that header statement skipped. All the statements following this header are related to the previous header which leads to more problems since statements which can only be associated to a PROCESS are being attributed to a GROUP name. More errors will occur from this resulting; hence, the PROCESS, RCVS and SUBPARTS statements will not be entered into the data base. To correct this error, the statements that were omitted could be entered by another INPUT-PSL command. A far more serious problem occurs if the "previous" header was also a PROCESS. For example:

```
PROCESS PX;
RCVS: I1, I3;
GENS: G1, G2;
PROCESS P1;
RCVS I1, I2;
SUBPARTS P2, P3;
ELEMENT E1, E2;
```

If this were the case, then only one error would be caught by URA (the misspelled "PROCESS") and the following RCVS and SUBPARTS statements would be attributed to PROCESS PX. If this mistake were discovered, the user would have to delete the two statements from PX and then reinput the information for P1:

```
DELETE-PSL
PROCESS PX;
RCVS I1,I2;
SUBPARTS P2,P3;
EOP
INPUT-PSL UPDATE
PROCESS P1;
RCVS I1,I2;
SUBPARTS P2,P3;
EOP
```

ii) Illegal Statement

This error is designated by the URA error:

```
URA266:ILLST: ILLEGAL STATEMENT IN THIS SECTION
```

This error can be caused simply by using a statement that is not allowed for that particular section. Using a CONSISTS statement in a PROCESS section would obviously generate this error. The other case occurs when an error is made in a header section statement and all the following statements might be incompatible with the previous header section name. Whenever this error is encountered, the statement is not put into the data base.

iii) Illegal Header Statement

If an error occurs in a header statement and URA is able to identify it as a header statement, the following error will be given:

URA025:HEAD: INVALID HEADER STATEMENT-STATEMENTS WILL BE IGNORED

This means that all the statements up to the next header statement will be ignored and not input into the data base. All the statements ignored must be reinputted using another INPUT-PSL command to be put into the data base.

iv) Input Line Too Long

If a number of URL statements are used on one line of the input file or if the URL statement is very long, it may run over the 72 column restriction. Should this occur, usually URA010 or URA011 will be generated specifying that improper syntax has been encountered. Note that no error message is generated for the fact that the statement runs over the 72 column restriction. Errors are encountered because anything over column 72 is ignored. Therefore, names may be truncated or a semicolon lost.

v) Name Too Long

It is an easy thing to mistake a 31 or 32 character word for 30 characters. Names longer than 30 characters are caught by URA and flagged by the error:

URA002:NLEX: NAME TOO LONG

The statement that used the name is still entered into the data base but the name is stored in a truncated form in the data base. If the truncated form of the name is not satisfactory, it is a simple matter to change the name via the RENAME command.

vi) Using URA Reserved Words Incorrectly

Most syntax errors are fairly easy to detect; a misspelled word, improper format, etc., but one of the hardest to detect is the improper use of a URL reserved word. For example, the following statement would be flagged by a URA010 or URA000 error message as having a syntax error:

ATTRIBUTE TYPE A;

The letter "A" happens to be a URL optional word and cannot be used as a user-defined name. Detecting these reserved words can get trickier than this, however, as the statment:

PROCESS D,F,G,K;

seems correct, but "F" is the abbreviation of the URL reserved word "FALSE." The key to finding these errors is to watch where the "$" character is printed by URA. It is usually printed directly after the location of the source of the error. The statement is ignored should this type of error occur and the only solution is to reinput the data using a different name. A list of all URL reserved words is given in Appendix B of the "User Requirements Language, Language Reference Manual."[1]

vii) Synonym Too Complicated

This error is specified by the URA error:

URA206:SETSYN: UNABLE TO MAKE SYNONYM-TOO COMPLICATED

This is caused by specifying various relationships about two names and then attempting to make one a SYNONYM of the other. The problem lies in combining these relationships. The statements:

```
GROUP G1;
USED BY P2;
PROCESS LONG-PROCESS-NAME;
SUBPARTS P3,P4;
SYNONYM P2;
```

will generate the error. P2 is implicitly defined to be a PROCESS just in the context in which it is used in the second statement; it also has a relationship with G1. Now LONG-PROCESS-NAME is defined and has relationships formed with P3 and P4. In the last statement, an attempt was made to make P2 and LONG-PROCESS-NAME the same PROCESS and the error is generated. The whole problem could have been avoided if the user had maintained the convention of issuing SYNONYM statement directly after the header statement as shown below:

```
GROUP G1;
USED BY P2;
PROCESS LONG-PROCESS-NAME;
SYNONYM P2;
SUBPARTS P3;P4;
```

If the statements had been inputted in this manner, LONG-PROCESS-NAME would not have had any relationships formed with other names (P3 and P4 in the previous example) and the assignment of P2 as a SYNONYM would have been successful.

---

[1] Part II, URL User's Manual.

Since the error does occur, there exists a method of correcting this problem:

1) Retrieve all information for one of the names via the PUNCH parameter for the PPS command.

2) Delete the name for which the information was retrieved from the data base.

3) Alter the PUNCH information so that all the information now pertains to the name still in the data base.

4) Enter the modified PUNCH information as input to the INPUT-PSL command.

In this way, all information about P2 is given to LONG-PROCESS-NAME and P2 is assigned as a SYNONYM if future references to the name are necessary. It is much easier to maintain the convention of assigning SYNONYMS directly after the header statement.

viii) Names Used in Wrong Context

This type of error accounts for the majority of the error messages presented in Section 9.

URA202:NLIST: NAME PREVIOUSLY USED DIFFERENTLY-IGNORED

is an example of diagnostics presented for this type of error. The statement will be ignored and the only way to resolve the problem is to reinput the information in an acceptable format.

ix) Breaking Section/Statement Rules

Several error messages can be generated by attempting to break the rules set forth in the "User Requirements Language, Version 4.0, Language Reference Manual,"[1] for statements within a particular section. In using the PART statement, for example, an object may be PART of only one object and failure to comply with this rule will result in:

URA061:RWLIST: ALREADY PART OF SOMETHING ELSE

or some analogous URA error message. These error checks are made to enforce the rules set forth in the "Language Reference Manual" and ensure that the problem statement is still meaningful. Other messages presented for this type of error are:

---

[1] Part II, URL User's Manual.

URA214:OTHERS: CONNECTIVITY ALREADY GIVEN FOR THIS RELATION

or,

URA060:APPLES: SECOND MAILBOX FOR PD ILLEGAL

If the user wishes to replace the information stated in the data base, e.g., replace the MAILBOX for a problem definer, the relationship should be deleted via DELETE-PSL and then the correct information should be inputted using the INPUT- PSL command.

10.2 Logical Errors

These errors occur when inputting information into the data base (as input errors do), but no diagnostics are given in the AS-IS SOURCE LISTING. These errors might be detected by scanning the complete list of names in the data base (NAME-GEN) and the complete problem statement (FORMATTED-PROBLEM-STATEMENT). These errors can also be detected when reviewing the contents of any of the other reports available on URA.

Misspelled Names

A simple spelling error can result in two names which look very similar, but which are treated as two different objects in the data base.

For example, if the name, "CALENDAR-DAY" was used to specify a particular INTERVAL in the data base and then "CALENDAR-DAYS" is used in the statements.

```
INTERVAL: CALENDAR-week;
CONSISTS: 7 CALENDAR-DAYS;
```

the two names become completely different objects (to URA). URA does not know that the two are the same object and it is up to the user to detect and correct this mistake which can be done in the following manner.

1) Retrieve all information for one of the names via the PUNCH parameter for the FPS command.

2) Delete the name for which the information was retrieved from the data base.

3) Alter the PUNCH information so that all the information now pertains to the name still in the data base.

4) Enter the modified PUNCH information as input to the INPUT-PSL command.

The effect of doing this for the previous example would be that:

i) All information given about CALENDAR-DAYS is transferred to CALENDAR-DAY and then CARLENDAR-DAYS is deleted from the data base. If it is desirable to use the plural form of the name in the data base then it should be a SYNONYM, this can be done in a DESIGNATE statement:

```
INPUT-PSL
DESG CALENDAR-DAYS SYNONYM CALENDAR-DAY;
EOF
```

ii) Since names can consist of letters or numbers, and certain special characters, another common misspelling error is to substitute the letter "O" for the number "0". It is often very difficult to detect this and so there appear to be two names, spelled exactly the same in the data base. This can be corrected in the same way as the previous problem.

iii) When the spelling error only involves one name (if TIME-CARD was spelled TIMECARD in all instances) then this problem could easily be solved by using the RENAME command:

```
RENAME O=TIMECARD N=TIME-CARD
```

If both TIME-CARD and TIMECARD are defined in the data base, then the same procedure used to change CALENDAR-DAYS must be performed.

Redundant Objects

Another error which occurs quite frequently is to define one object by two different names, not realizing that they are representing the same thing. EMPLOYEE-RECORD and EMPLOYEE-DATA may be defined separately in the data base, but represent the same thing. To resolve this redundancy, the information for the two names must be combined. This can be done in the same manner as given for correcting the misspelled names (involving two names).

Missing Semicolons

Most often, a missing semicolon will be detected as a syntax error (as described in section 10.1). There is one particular case where a missing semicolon would not generate any error message:

```
PROCESS P1;
DESCRIPTION;
     THIS IS A DESCRIPTION COMMENT ENTRY THAT IS MISSING
     THE SEMICOLON.
RCVS I1,I2;
```

GENS 01,02;

What happens here is that the RCVS statement becomes part of the DESCRIPTION comment entry. A semicolon was omitted in terminating the lines intended to be the comment entry, but URA simply searches for the first semicolon to signify the end of the comment entry. To solve this problem the DESCRIPTION statement must be replaced and the RCVS statement must be added to the data base. This can be accomplished by the following procedure.

1) Generate the incorrect comment entry in the form of PUNCH information (via the PUNCH-COMMENT-ENTRY command).

2) Alter the PUNCH information so that the comment entry is correct.

3) Use the modified PUNCH information as input to the REPLACE-COMMENT-ENTRY command.

4) Add the RCVS statement via an INPUT-PSL command.

Correctness and Completeness

For the most part, it is up to the problem definer to maintain correctness of the problem statement and URA maintains correctness of the data base. The problem definer has the ability to do this through usage of the DELETE-PSL and INPUT-PSL commands. Completeness can also be determined by the problem definer or improved through use of the INPUT-PSL command.

PART II

USAGE OF THE USER REQUIREMENTS ANALYZER UNDER MULTICS

1. INTRODUCTION

URA extracts information from URL statements and stores it in a URA data base. Once this information (a requirements statement) is in the data base, it can be modified, new information can be added to it, and reports can be generated presenting the status of the requirements statement. These actions are implemented by the URA commands available in the URA processing mode.[1] This mode of operation may be attained by accessing the URA software available under Multics. Therefore, by understanding the Multics commands that aid in interacting with URA, and the URA command language, the URA user can effectively manipulate the contents of a URA data base.

The format of this part serves an important purpose. The first five sections deal with Multics and URA at an introductory level. Section 2 presents necessary information for reference to Multics operating system. Section 3 explains the procedure of accessing URA once on Multics. Sections 4 and 5 present practical concepts and conventions which aid in using URA under Multics. Once access to URA has been achieved, Sections 6, 7 and 8 present the manner in which Multics interacts with the various commands. Several examples are given in those sections in order to better illustrate the results of specific implementations. Section 9 deals with using Multics to handle errors encountered in the use URA. In this chapter, the Multics naming conventions are used, hence, all Multics and URA commands appear in lower case.

2. USING MULTICS

An excellent tutorial on the use of Multics can be found in the Multics User's Guide Honeywell Order Number AL40 and reference material will be found in the Multics Programmer's Manuals. The remaining sections in this chapter assume a familiarity with the Multics command language.

3. THE URA ENVIRONMENT

Assuming that the user is already logged into Multics, he should first gain access to all URA software which is contained in the URA directory. The user may do this by adding the URA directory to his search rules after his working directory. The following command shoud be used:

```
asr >ml>CARA -after working_dir
```

---

[1] A processing mode is defined by the software system in control when any type of command is issued. The debug, facility, the editor, URA, etc. all define processing modes and thus, a particular set of commands.

A URA data base with the name uradb may be initialized by the following command:

```
exec_com >ml>CARA>initdb uradb
```

This will create an empty initialized data base of the default size in the current working directory. To create a data base with a non-default size see Appendix B for creating data bases.

Once a data base has been initialized, the user may enter URA mode with the Multics command:

```
exec_com >ml>CARA>ura
```

This command is used whenever the user wishes to enter URA mode, whereas a data base need only be initialized at the beginning of a project. Should a user desire to expand or reorganize his data base, the dump/restore programs described in Appendix C may be used.

4. SPECIFYING INPUT TO URA COMMANDS

This section specifies information relavant to using and manipulating Multics segments to be used as input files to URA commands. The "input" and "file" parameters of URA commands may be used with standard Multics segments or input may be specified as coming from the terminal in an interactive mode.

4.1 Entering Data into a Data Set

Using one of the Multics editors, the user can create segments to be used with the "input" and "file" parameters. Such segments may contain names, URL statements, etc.

If the user has URL statements in the segment named mos.input.psl in the current working directory, he may use it as input for the ip command as follows:

```
ip input=mos.input.psl update
```

4.2 Specifying Input Data Interactively

It is usually advantageous to enter data into segments before it is used as input to URA as the user may correct errors and omissions without difficulty using the Multics editor. There are times when the user may wish to enter data directly to the Analyzer. In such cases, the user specifies "term" instead of the segment name. For example, to provide input via the terminal for the ip command, use:

```
ip input=term update
```

5. RECEIVING OUTPUT FROM URA COMMANDS

This section specifies information relevant to using and manipulating Multics segments to be used as output for URA commands. In Multics, output files are specified in the output parameter of the URA set command, and via the punch parameter for other URA comands.

5.1 Using the Output Parameter

The output parameter in the URA set command allows the problem definer to specify where all reports generated by URA commands are to be printed. If nothing is specified, all output is sent to the terminal. To specify that output is to be sent to a segment:

```
set output=segment-name
```

From this point, all reports generated by URA will be written into this segment. The report output may be reassigned to the terminal by:

```
set output=term
```

5.2 Using Punch Segments

For some commands, a punch file is produced. If the user does not explicitly assign the punch file, a default segment name will be used. These default segment names are given in Appendix E.

6. CONTROL COMMANDS

These commands control the operation of URA in some way without actually causing any output themselves.

6.1 Set Command

The most common use of this command is to change the default data base, or to reroute the report output. To change the data base to be used on subsequent commands:

```
set db=data-base-segment-name
```

5.2 Display Command

This command displays the current settings of global switches and parameters set using the SET command or merely defaulted to.

For example, if the user wishes to display the settings of all switches and parameters, he should type:

```
display all
```

To check the current data base setting, he should type:

```
display db
```

6.3 Stop Command

This command returns the user to Multics command mode. All parameters changed via the URA set command return to their default values.

7. MODIFIER COMMANDS

```
change-type
delete
delete-comment-entry
delete-psl
input-psl
punch-comment-entry
rename
replace-comment-entry
```

8. REPORT COMMANDS

```
consist-comparison
consist matrix
contents
data-process
dictionary
entity-identifier
extended-picture
formatted-problem-statement
frequency
kwic
name-gen
name-list
picture
print-attribute-values
process-chain
process-input-output
punch-comment-entry
```

```
resource-consumption-analysis
security-consistency-analysis
structure
summary
```

9. ERROR CONDITIONS

In addition to error comments generated by the URA system, there are occasional errors associated with the interaction between the URA system and Multics.

9.1 Initial Messages

When the user first enters URA mode, he may receive messages from Multics that "io-switch does not exist." These may be ignored.

9.2 Abnormal Termination

If any of the commands is unable to terminate in the normal manner, the Fortran monitor may be entered before URA has had a chance to close all files. The Fortran monitor will ask the user if these files are to be closed. In all cases, the user should answer "yes."

9.3 Data Base Already Open

It is possible for the user to get into a state where URA cannot open his data base because it thinks that it is already open. If this happens, the user should return to Multics mode, then issue the Multics command:

```
ml>CARA>closeit
```

PART III

URA OUTPUTS

1. INTRODUCTION

Once a URL description of an information processing system has been entered into a URA data base the user has the option of retrieving the stored information in several different standard formats called URA reports. Each URA report has particular characteristics with respect to its purpose, the amount of retrieval and analysis required to generate the report, the information presented in the report, the format, etc. In this sense, each report can be classified by its characteristics to provide an overall description of the report and to aid in determining how the report may be used to aid problem definers in checking the validty of the URL description and to improve on its completeness.

Only the reports generated by report commands will be presented in this paper. The reports generated by modifier commands are described in the "User Requirements Analyzer User's Manual," Part I. The manner of specifying input to report commands is given in Section 4 of Part I and Section 4 of Part II. The manner of receiving output from report commands is given in Section 5 of Part I and Section 5 of Part II.

Section 2 of Part III presents the objectives of URA reports with respect to those relationships to the logical system design process and their advantages in the system documentation procedure. Section 3 describes the manner in which information is extracted, from a URA data base to produce reports.

Section 4 presents several categories for classifying and describing URA reports based on contents, format and usage. Section 5 consists of descriptions of each of the standard URA reports available in Version 3.2 of URA.

2. OBJECTS OF REPORTS FROM A URA DATA BASE

2.1 Purpose Of URA Reports

The purpose of URA Reports is to present information retrieved from a URA data base (which contains the description of a particular system) in a format which is useful to persons who are documenting the target system, who desire to understand the target system, or who are involved in the design of the target system.

This requires that the reports must be of various formats, contain various types and levels of information, and be oriented at the various types of user's.

With respect to those persons documenting the target system, the reports must present information resulting from modifications to the URA data base. (These reports are presented in Part I.) In

addition, reports must present the status of the URA data base after modifications have been made (i.e., successful modifications). Those reports are basically the same as those used by people who desire to understand the target system. They present selected portions of the information in the data base in various formats. A few particular reports may also be used to aid those persons designing the target system. They usually present the results of extensive analysis on information in the data base.

2.2 Relation of URA Reports to Logical System Design

There are two major objectives in logical system design:

- To produce a proposed system that is the best possible in terms of what it will cost to build, cost to operate and what it will contribute to the organization.

- To minimize the cost and time to produce this "optimum" target system.

The goal of developing computer-aided methods for use in logical system design is to contribute to the above objectives. At the present time, it is not possible to achieve an optimum solution for both of these objectives. One contribution that can be made by a computer-aided method is to improve the "quality" of the description of the target system. Quality is defined in terms of consistency, unambiguity and completeness.

Consistency means that no statements made in the description contradict, or are incompatible with, other statements and any particular object is referred to by the same name throughout the description.

Unambiguity means that statements and relationships are made so precisely that interpretation is uniform by all readers.

Completeness means that all necessary relationships are given and no objects have been omitted from the description.

The quality objectives can be aided at three levels:

1) URA will enforce consistency and unambiguity through the syntax analysis and reference checks made when data is entered into the URA data base.

2) The URA reports will make it easier for the problem definer to detect logic errors in the problem statement, unresolved conditions, etc.

3) Some reports are available to the problem definer to aid in detecting incompleteness and inconsistencies of the problem statement

Therefore, the first objective in logical system design can be attained by improving the quality of the documentation.

The second objective, that of minimizing design cost and time, is aided by transferring much of the clerical workload to the computer. The Analyzer, URA, maintains an up-to-date record of all information collected. The preparation of this information for use by analysts,[1] management, etc., can be readily retrieved at request.

To meet these objectives, URA offers three classes of outputs:

- Reports to aid the analyst.

- Reports to aid the project management.

- Reports to aid the designer.

- Final specifications.

Reports to aid the analyst are basically those which aid in resolving inconsistencies, ambiguities, and incompleteness in the logical system design problem statement.

The reports of concern to project management pertain to status of the project in the form of amount of information entered into the URA data base, etc. The reports which are of benefit to the designer are those which reflect inconsistencies and incompleteness in the problem statement which must be resolved in order to develop a design for the proposed system. They also present information in a manner that optimal or feasible designs may be formulated.

The final specifications are the end result of the logical system design phase using URL/URA. They express all the information in the URA data base in an easy-to-read format. These specifications consist of a series of URA reports generated in a particular order and format.

2.3 Advantages of Using URA Reports

In contrast to manually produced documentation, (i.e., handwritten or typed, the URA reports have several advantages with respect to maintainability, format, usefulness, etc.).

---

[1] NOTE: Throughout Part III, the terms analyst and problem definer are used synonymously. The problem definer is a person responsible for writing a system description (or part of one) in URL. A URA user is a person who uses the URA software to update or retrieve information from a URA data base.

Maintainability

All changes made to the URL description of an Information Processing System are made via the URA data base. All documentation (URA reports) produced from the information in the data base after a change is up-to-date. There is no need to change previous documentation.

Changes made to manually maintained descriptions usually require modification of all existing sets of documentation, and modification of each portion of the documentation affected by the change. This process can cause serious errors in the documentation or require extensive rewriting (and retyping) of the previous documentation.

Format

Each URA report is formatted according to the purpose of the report, taking into consideration the orientation of persons intended to use the report. Therefore, information consisting of many complex relationships may be presented in a graphical format, to make the information easier to interpret. Likewise, a matrix format may be used to present a large amount of information at one time. Formats are standardized so that interpretation is uniform by all users.

Formats of manually produced documentation are often not standardized leading to problems and conflicts in interpretation. Presentations of large amounts of information on a few pages is often a problem because of difficulties in completion and keeping the information up-to-date.

Usefulness

Each URA report has been designed for a particular purpose, i.e., to meet the system documentation needs of one or more users. In an effort to maintain this, all information in the reports is presented in a well-structured manner and the reports are designed to be consistent in their presentations.

Many manually produced documents are done ad hoc with possibly no serious considerations on how the documents may be used to benefit the system building process. In addition, the documentation is often difficult to use because of inconsistencies in the manner in which information is presented and seemingly lack of organization.

Availability

Documentation can be produced anytime (on request) once

information has been stored in the URA data base. This reduces the lag time usually encountered in the manual production of documentation. This permits important decisions to be made when the problem is encountered rather than "next week, when all the information is available." Physical production of the reports is very fast since this is usually done using a line printer or terminal.

Too often the documentation is produced after the fact. This is especially common if the people are very technically inclined. They would rather sit at the terminal or write program code rather than write documentation. Therefore, when documentation is needed, little is available or the organization of the information makes it very difficult to use. Physical production of the documentation then requires many hours of manual labor behind pencils and typewriters excluding the time and costs of reproducing the original. (It is important to note that in most instances that documentation produced in this way is usually out-of-date by the time it leaves the typewriters.)

Selectivity

URA reports may be generated containing all the information known about the target system as containing specific information about one particular name relevant to the description. This capability allows the user to see only what is desired rather than getting too much or too little.

It would be virtually impossible to incorporate all possible combinations of information in a manually produced system in an effort to anticipate any type of request. Therefore, the description of the target system is presented in a select number of ways and any information not directly available must be desired from the available information (assuming it is in some way desirable).

Analysis

URA offers reports which aid in checking completeness and consistency of the problem statement as it exists in the URA data base. Many of the reports present information in a manner which allows visual-analysis of the information to check for these qualities. Other reports incorporate these checks as part of the information presented in the report and is referred to as computer-aided analysis. (Visual analysis implies that the checks are made by the user where computer-aided analysis is performed by the computer.)

In most forms of manually produced documentation the formats used make it difficult to check for completeness and inconsistencies in the documentation. Inconsistencies, in particular, go undetected very easily as a result of various

spellings of the same name in the description. Where one person may know that all the versions of the name refer to the same thing, others may not.

Extensions

In addition to the standard reports available in a particular version of URA, the users may write their own programs which generate reports to present information particular to their applications of URA. The new reports can be incorporated into any future documentation packages.

Manually produced documentation may incorporate complex reports involving a great amount of computation and analysis in its production, but this requires the same amount of computation and analysis anytime the report is to be presented. Rather than consuming the analyst's time in producing the report, it is more feasible to let the computer do it (and in less time).

3. GENERATION OF REPORTS

All URA reports are produced by issuing commands to URA. All commands available for URA and the reports generated by each are given in "User Requirement Analyzer Command Descriptions." Descriptions of each report and the name(s) of the command(s) used to generate it are given in Section 5 of Part III.

3.1 URA Command Language for Reports

The programs which are initiated by a particular report command, retrieve information from the URA data base and output it in some meaningful format. No modifications are made to the information in the data base; their sole function is to retrieve the information and display it in some manner. In the process of collecting and displaying the information analysis may be done and the results printed as part of the report.

Most report commands have one or more parameters that may be used in conjunction with the command for one of the following purposes:

- To specify data to be used as input to the command (Input data parameters).
- To specify how data used as input is to be interpretted (Input control parameters).
- To specify what options the user has in generating the report with respect to content (Output option parameters).
- To specify what options the user has in generating the report

with respect to format (Output format parameters).

The manner of specifying commands and their parameters is given in Part IV and the manner of using the commands and parameters is given in Parts I and II.

3.2 Retrieval of Information from the URA Data Base

There are basically three types of information stored in a URA data base.

1) Names and types of objects defined by the user.

2) Comment entries (narrative and free format descriptions of objects).

3) Connections among objects and between an object and comment entry.

The first type of information is stored as name records, the second as comment entry records, and the last as NUB records.[1]

Each URA report presents information taken from one or more of these different types of records. Table 2 gives the information presented by each report. In addition each URA report may also present information derived from the information in one or more of these different types of records.

The manner in which the information is presented differs from one report to another. The relationship between two name records may be either implied by the report format or explicitly defined. For example, the FORMATTED PROBLEM STATEMENT would present that "employee-address" CONSISTS of "street," "city," "state," and "zip-code." The CONTENTS REPORT, on the other hand would present this in the following format:

```
1 employee-address
  2 street
  2 city
  2 state
  2 zip-code
```

Therefore, there are three major aspects relative to what is contained in a URA report:

1) What type of information is presented:

   a) Names and types of object

---

[1] See the description of these records as presented in Section 7 of Part I for a better understanding of the data base structure.

b) Comment Entries
c) Connections among objects and between an object and comment entry

2) How the information is presented:

a) As taken from the URA data base
b) Derived from information in the data base

3) How Relationships are defined:

a) explicitly
b) implied

Report Name	Name Records	Comment Entries	NUB Records
ATTRIBUTE REPORT	Yes	No	Yes
CONSISTS COMPARISON MATRIX	Yes	No	Yes
CONSISTS MATRIX REPORT	Yes	NO	Yes
CONTENTS REPORT	Yes	No	Yes
DATA BASE SUMMARY[1]	No	No	No
DATA PROCESS REPORT	Yes	No	Yes
DICTIONARY REPORT	Yes	Yes	Yes
EXTENDED PICTURE	Yes	No	Yes
FORMATTED PROBLEM STATEMENT	Yes	Yes	Yes
FREQUENCY REPORT	Yes	No	Yes
IDENTIFIER INFORMATION REPORT	Yes	No	Yes
KWIC INDEX	Yes	No	No
NAME GEN	Yes	No	No
NAME LIST	Yes	No	No
PICTURE	Yes	No	Yes
PROCESS CHAIN	Yes	No	Yes
PROCESS INPUT/OUTPUT	Yes	Yes	Yes
PUNCHED COMMENT ENTRIES	Yes	Yes	Yes
RESOURCE CONSUMPTION ANALYSIS	Yes	No	Yes
SECURITY CONSISTENCY ANALYSIS	Yes	No	Yes
STRUCTURE	Yes	No	Yes

Table 2

Types of Information Taken and Presented by URA Reports

[1] All information in this report is "derived" from the information taken from each of the different types of records.

ATTRIBUTE REPORT

Purpose

This report is intended to present the system properties aspect of the target system description with respect to the ATTRIBUTES defined and used in the description.

Information Presented

The report presents, for each ATTRIBUTE name used as input, all names in the data base to which the ATTRIBUTE applies and the associated ATTRIBUTE-VALUES for the names. In effect, this presents information given by the ATTRIBUTE statement.

Format

Each ATTRIBUTE name is numbered, 1*, 2*, etc., as it is encountered as input and printed on the report. Each name to which a particular ATTRIBUTE applies is also numbered.

The URA statements:

```
INPUT: time-card;
   ATTRIBUTE arrival-type scheduled;
```

would be presented as:

```
1* ATTRIBUTE: arrival-type

          APPLIES TO:    VALUE:
        1 time-card      scheduled
```

in the ATTRIBUTE REPORT for the ATTRIBUTE arrival-type. Given a particular ATTRIBUTE name, the software generating the report searches the data base for all names the ATTRIBUTE APPLIES to (is connected to) and lists them under the "APPLIES TO:" heading in the report with corresponding ATTRIBUTE-VALUE under the "VALUE:" heading.

The format for the ATTRIBUTE REPORT is considered to be a list format.

Option and Alternatives

The report may be generated for a single ATTRIBUTE name (via the NAME parameter) or for a collection of ATTRIBUTE names specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input in generating the report must be checked that it is an ATTRIBUTE name before further processing continues. The name is then printed on the report. If the input name is not an ATTRIBUTE the message:

```
URAC93:MAINPAV: NAME HAS NO USAGES AS ATTRIBUTE FOR ANYTHING
```

is printed. Names and corresponding ATTRIBUTE-VALUES are then retrieved pair by pair and listed under the given ATTRIBUTE until no more pairs are found for the ATTRIBUTE.

The next ATTRIBUTE name from the input stream is taken (if there is one) and the process repeats.

Usages

In many applications of the use of the Language and Analyzer, certain ATTRIBUTE names may be designated as mandatory for a description. For example, some applications may require that every PROCESS defined as part of the description must be specified as being either manual or automated via the ATTRIBUTE process-type. Generation of the ATTRIBUTE REPORT for the name process-type allows the analyst to determine if the current description is accurate, complete and consistent in this respect.

The report also presents those names which are logically related because of common ATTRIBUTE-VALUES among them. In the previous example, there were two logical groups, manual and automated. In practice there are usually several possible ATTRIBUTE-VALUES.

Examples

Figure 19 presents the output resulting from generating the ATTRIBUTE REPORT using the names produced by NAME-GEN as input. The following Analyzer commands were given:

```
NAME-GEN S='ATTRIBUTE'
PRINT-ATTRIBUTE-VALUES
```

FIGURE_19

ATTRIBUTE REPORT

PARAMETERS FOR: PAV

FILE

1* ATTRIBUTE: arrival-type

	APPLIES TO:	VALUE:
1	time-card	scheduled
2	salaried-employment-form	random
3	hourly-employment-form	random
4	tax-withholding-certificate	random
5	employment-termination-form	random

2* ATTRIBUTE: color

	APPLIES TO:	VALUE:
1	paper	white

3* ATTRIBUTE: complexity-level

	APPLIES TO:	VALUE:
1	salaried-employee-processing	high
2	hourly-employee-processing	high
3	new-employee-processing	medium
4	terminating-emp-processing	low

4* ATTRIBUTE: copies

	APPLIES TO:	VALUE:
1	hourly-employee-report	3
2	salaried-employee-report	3
3	terminated-employee-report	2

FIGURE_19

ATTRIBUTE REPORT

4	hired-employee-report	2

5* ATTRIBUTE: data-standard

	APPLIES TO:	VALUE:
1	birthdate	date
2	current-date	date
3	employment-date	date
4	pay-date	date
5	termination-date	date

6* ATTRIBUTE: number-of-lines

	APPLIES TO:	VALUE:
1	page	20

7* ATTRIBUTE: occurrence-type

	APPLIES TO:	VALUE:
1	validation	unscheduled

8* ATTRIBUTE: processor-type

	APPLIES TO:	VALUE:
1	validation-clerk	human

9* ATTRIBUTE: type

	APPLIES TO:	VALUE:
1	employee-identification-number	numeric
2	department	numeric

FIGURE_19

ATTRIBUTE REPORT

3	job-number	numeric
4	pay-grade-code	numeric
5	salary	numeric
6	total-hours	numeric
7	number-of-deductions	numeric
8	job-title	character
9	supervisor	character
10	city	character
11	state	character
12	street	character
13	employee-name	character

CONSISTS COMPARISON REPORT

Purpose

To present data structure information for SET, INPUT, OUTPUT, ENTITY and GROUP names given as input. The report by-passes all intermediate levels of data structure and only presents the lowest level constituents of those names given as input.

Information Presented

Structure information based on CONSISTS statements in the data base can be presented for SET, INPUT, ENTITY and GROUP names given as input. However, rather than presenting all levels of the data structure, only the highest and lowest levels are present ignoring all other levels. For example, the structure:

```
1 hourly-employee-record
  2 employee-name
    3 surname
    3 initial
    3 first-name
  2 employee-identification-number
  2 social-security-number
```

might appear in the CONTENTS REPORT which presents all levels of the data structure for hourly-employee-record. The CONSISTS COMPARISON REPORT, however, would present the names:

```
surname
initial
first-name
employee-identification-number
social-security-number
```

as the low level components of the data structure for hourly-employee-record.

In addition to the data structure relationships presented, similarities among data structures are identified and summarized in the report.

Format

The first part of the report specifies those names given as input, but do not have data structure relationships (CONSIST statements). Two lists are given, one giving all names given as input (and having CONSISTS information), and the other giving all the low level constituents of those names in the first list. Next comes the BASIC CONTENTS MATRIX where each row of the matrix represents one of those names given in the list of low

level constituents and each column of the matrix represents one of the names given in the list of input names. All names in the list and rows and columns are numbered so that the correspondence between a particular row or column and the name that it represents is one to one. A relationship between a name represented by a particular row and a name represented by a particular column is designated by an asterisk entry (*) at the intersection of the row and column in the matrix. A blank entry designates that no such relationship exists.

A second matrix called the CONTENTS SIMILARITY MATRIX is also generated to present similarities in the data structures (for those names given as input) and represented in the BASIC CONTENTS MATRIX. All information in the CONTENTS SIMILARITY MATRIX is derived from information presented in the BASIC CONTENTS MATRIX. All the names used as input are represented by a column number and row number in the matrix. The numbers are the same so that any given object is represented by row I and column J. The matrix should be read from row to column as saying: the data object represented by row I has an integer number of low level constituents in common with the data object represented in column J. When I=J, the number of low level constituents of any object in common with itself is presented. This is, of course, the total number of constituents for that given data object. The final section of this report is the CONTENTS SIMILARITY ANALYSIS which presents those input names that have identical lowest level constituents or which are strict subsets (at the lowest level) of other input names.

Options and Alternatives

No options are available to change format or content of the report.

Analysis

Each name given as input is searched for in the Analyzer data base. If the name is not found, the message:

URA271: MAINCNC: NAME NOT IN D.B.-

is printed and is not represented in the matrices.

For each name defined in the data base with CONSISTS information, its components are then found via the CONSISTS statement. If its components have CONSISTS information then the procedure is continued until only ELEMENTS are encountered (which may not have any sub-components) or no more CONSISTS information can be found. The lists of input names and low level constituents are then printed on the report.

The BASIC CONTENTS MATRIX is then printed out to illustrate the

relationships between the names in the two lists and each relationship is designated by an asterisk.

The CONTENTS SIMILARITY MATRIX is produced by counting the number of column entries in common between any two rows of the BASIC CONTENTS MATRIX. The diagonal is produced by counting the total number of asterisks in the BASIC CONTENTS MATRIX for a given row.

The CONTENTS SIMILARITY SUMMARY is produced by inspecting the numerical values in the CONTENTS SIMILARITY MATRIX. If a particular number presented in the diagonal occurs elsewhere in the same row of the matrix, it means that a particular name (represented by the row) has all of its constituents, in common with another name. If the other name has the same number of constituents, then the data structures are identical. If the other name has more constituents, then the first name has a data structure which is a subset of the others.

Usage

One use of the CONSISTS COMPARISON REPORT is to detect redundant or similar data structures. Its ability to do this lies in its presentation (ignoring all intermediate levels of data structure).

This aids the analyst in detecting possible errors in the target system description and simplifying it should similarities in structures occur. This is usually beneficial when the report is generated for all INPUT, OUTPUT and ENTITY names in the description as input.

The report is also beneficial to the system designer who may utilize it to optimize structures that the software will eventually have to access. Identical or similar structures for different ENTITIES, for example, may be mapped into the same type of storage structure to reduce complexity of the software.

Examples

Figure 20 presents the results of generating the CONSISTS COMPARISON REPORT for all INPUT and OUTPUT names in a particular data base. The following commands were used to generate the example:

```
NAME-GEN S='INPUT OR OUTPUT'
    CONSISTS-COMPARISION
```

Note that the two names employee-information and paysystem-outputs are not included in the matrices because they do not have CONSISTS information.

FIGURE_20

CONSISTS COMPARISON REPORT

PARAMETERS FOR: CNC

FILE

URA273:CNCBLD : NAME DOESNT CONSIST OF ANYTHING - employee-information
URA273:CNCBLD : NAME DOESNT CONSIST OF ANYTHING - paysystem-outputs

FIGURE_20

CONSISTS COMPARISON REPORT

BASIC CONTENTS MATRIX

The rows are the given input names.

The columns are the lowest level objects which are contained in the rows, with intermediate groups ignored.

If any columns are group names, then the definition is incomplete.

If any columns are ambiguous names, they are possible elements.

	ROW NAMES	
1	employment-termination-form	INPUT
2	hourly-employment-form	INPUT
3	salaried-employment-form	INPUT
4	tax-withholding-certificate	INPUT
5	time-card	INPUT
6	error-listing	OUTPUT
7	hired-employee-report	OUTPUT
8	hourly-employee-report	OUTPUT
9	pay-statement	OUTPUT
10	salaried-employee-report	OUTPUT
11	terminated-employee-report	OUTPUT

	COLUMN NAMES	
1	surname	ELEMENT
2	initial	ELEMENT
3	first-name	ELEMENT
4	social-security-number	ELEMENT
5	termination-date	ELEMENT
6	employee-identification-number	ELEMENT
7	employment-status	ELEMENT
8	sex	ELEMENT
9	birthdate	ELEMENT
10	house-number	ELEMENT
11	street	ELEMENT
12	apartment-number	ELEMENT
13	city	ELEMENT
14	state	ELEMENT
15	zip-code	ELEMENT
16	phone	ELEMENT
17	job-title	ELEMENT
18	pay-rate	ELEMENT
19	current-date	ELEMENT
20	employment-date	ELEMENT
21	job-number	ELEMENT
22	pay-grade-code	ELEMENT
23	supervisor	ELEMENT
24	department	ELEMENT

FIGURE_2D

CONSISTS COMPARISON REPORT

BASIC CONTENTS MATRIX

ROW NAMES	COLUMN NAMES	
25	number-of-deductions	ELEMENT
26	status-code	ELEMENT
27	pay-date	ELEMENT
28	regular-hours-worked	ELEMENT
29	overtime-hours-worked	ELEMENT
30	hours-per-day	ELEMENT
31	error-code	ELEMENT
32	gross-pay	ELEMENT
33	total-hours	ELEMENT
34	check-number	ELEMENT
35	total-deductions	ELEMENT
36	net-pay	ELEMENT
37	federal-tax	ELEMENT
38	state-tax	ELEMENT
39	fica-tax	ELEMENT
40	cumulative-gross-pay	ELEMENT
41	cumulative-tax-deductions	ELEMENT
42	cumulative-fica-deductions	ELEMENT
43	cumulative-hours	ELEMENT
44	cumulative-state-deductions	ELEMENT
45	cumulative-federal-deductions	ELEMENT

FIGURE_20

CONSISTS COMPARISON REPORT

BASIC CONTENTS MATRIX

An * in (i,j) means that column j is contained directly or indirectly in row i. The columns do not consist of anything further. Intermediate groups are ignored.

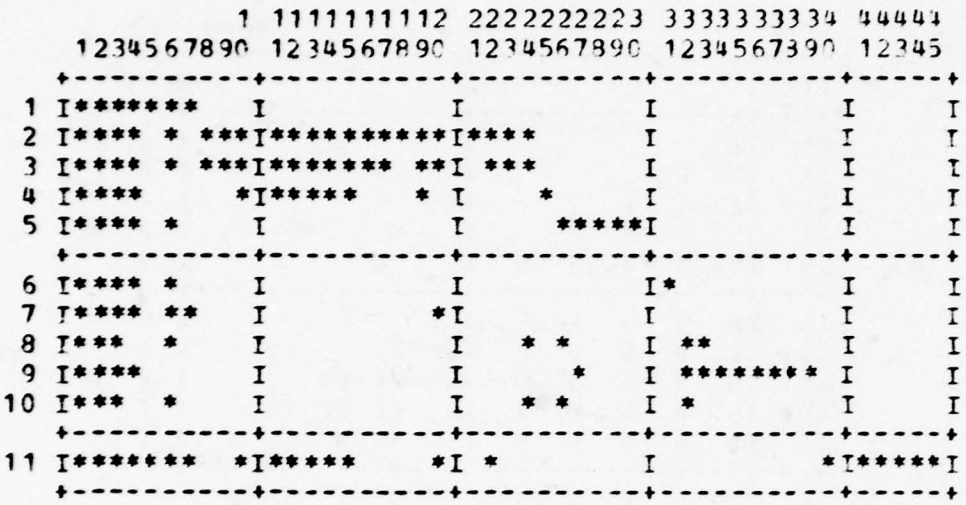


FIGURE_20

CONSISTS COMPARISON REPORT

CONTENTS SIMILARITY MATRIX

The number in (i,i) is the number of objects
at the lowest level contained in row i from above.

The number in (i,j) (i not equal j) is
the number of objects at the lowest level in
common between rows i and j from above.

	1	2	3	4	5	6	7	8	9	10	11
1	7	5	5	4	5	5	6	4	4	4	7
2		22	20	11	5	5	6	5	4	5	13
3			20	11	5	5	6	5	4	5	13
4				12	4	4	4	3	4	3	10
5					10	5	5	5	5	5	5
6						6	5	4	4	4	5
7							7	4	4	4	7
8								8	5	7	4
9									13	4	4
10										7	4
11											21

FIGURE_20

CONSISTS COMPARISON REPORT

CONTENTS SIMILARITY SUMMARY

ROW#	NAME		ROW#	NAME
1	employment-termination-form	IS A SUBSET OF	11	terminated-employee-report
3	salaried-employment-form	IS A SUBSET OF	2	hourly-employment-form
7	hired-employee-report	IS A SUBSET OF	11	terminated-employee-report
10	salaried-employee-report	IS A SUBSET OF	8	hourly-employee-report

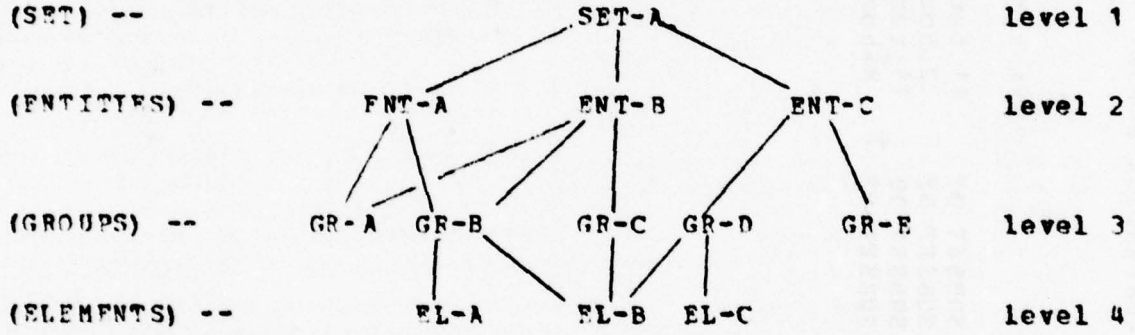
CONSISTS MATRIX REPORT

Purpose

To present the data structure (as specified by the CONSISTS statements), either above or below, each SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT name given as input is involved in.

Information Presented

For each name used as input (which must be a SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT) the report presents names the input name CONSISTS of (if the report is generated with the CONSISTS parameter in effect) or names the input name is CONTAINED in (if the report is generated with the CONTAINED parameter in effect). Take the following structure, for example, and assume that its description exists in an Analyzer data base in the form of CONSISTS relationships between the names:



If the CONSISTS MATRIX REPORT were generated for all the above GROUP names, GR-A, GR-B, GR-C, GR-D and GR-E, and the CONSISTS parameter was in effect when generating the report, the ELEMENT names EL-A, EL-B and EL-C would be presented in the report. The report would also designate GR-A and GR-E as not having any CONSISTS information available.

If the report was generating with the CONTAINED parameter in effect, and the same GROUP names as input, the ENTITY names ENT-A, ENT-B and ENT-C would be presented in the report

In essence, the CONSISTS MATRIX REPORT presents one level above or one level below designated starting points in the data structures described in an Analyzer data base.

The report could not be generated for SET names with the CONTAINED parameter in effect because the Language does not allow SETS to be CONTAINED in higher level structures. Likewise, the report could not be generated for ELEMENT names

with the CONSISTS parameter in effect because the Language does not allow ELEMENTS to CONSIST of lower level structures.

The report also presents statistics on the data structure information such as how many low level components a particular name consists of or how many structures it is contained in.

Format

If the CONSISTS parameter is used when generating the report, any names given as input which do not have CONSISTS statements as part of their descriptions are flagged at the beginning of the report. If the CONTAINED parameter is used, any names given as input which do not have CONTAINED statements as part of their descriptions are flagged.

Two lists of names are then presented, one labeled ROW NAMES and the other COLUMN NAMES. If the CONSISTS parameter was used when generating the report, the names designated as COLUMN NAMES are those which were given as input. If the CONTAINED parameter was used when generating the report, the names designated as ROW NAMES are those which were given as input.

In any case, each name under COLUMNS NAMES CONSISTS of zero or more names under ROW NAMES and each name under ROW NAMES IS CONTAINED in zero or more names under COLUMN NAMES. A matrix is then printed to show the relationships between the names designated as ROW NAMES (which are represented by the rows of the matrix) and the names designated as COLUMN NAMES (which are represented by the columns of the matrix). The rows and columns of the matrix are numbered to correspond to the number assigned to each name in the list of ROW NAMES and COLUMN NAMES, respectively.

An asterisk (*) entry at the intersection of a particular row and column of the matrix designates that the name represented by the row is CONTAINED in the name represented by the column.

Inspection of an entire row reveals all names that a particular name (represented by the row) is CONTAINED in. Inspection of an entire column reveals all names that a particular name (represented by the column) CONSISTS of.

A summary section is also included in the report presenting for each ROW NAME:

- The row it was represented by in the matrix (ROW).
- Its name type (TYPE).
- The number of * entries in its row (or the number of names CONTAINING it) (COUNT).

The summary presents for each COLUMN NAME:

- The column it was represented by (COLUMN).
- Its name type (TYPE).
- The number of * entries in its column (or the number of names it CONSISTS of) (COUNT).

The summary section for ROW and COLUMN names is ordered in decreasing order of COUNT.

Options and Alternatives

The report must be generated using either the CONSISTS or CONTAINED parameter. If the CONSISTS parameter is used, all names given as input must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. If the CONTAINED parameter is used, all names given as input must be INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names.

The report may be generated for a single input name (via the NAME parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input is searched for in the data base. If it is not found, the message:

URA098: CONCOL: NAME NOT IN D.B.-

is printed.

If the CONSISTS parameter is used, each name given as input must be checked that CONSISTS information is available for it. If no CONSISTS information is available, the name is listed under the message:

URA315:CONCOL : THE FOLLOWING DO NOT CONSIST OF ANYTHING:

The components of those input names which do have CONSISTS information are found. The list of all input names and the list of all components are then printed on the report.

If the CONTAINED parameter is used, each name given as input must be checked that CONTAINED information is available for it. If no CONTAINED information is available, the name is listed under the message:

URA311:CONROW : THE FOLLOWING ARE NOT CONTAINED IN ANYTHING:

Those names which the input names are CONTAINED in are found. The list of all input names and the list of names they are CONTAINED in are then printed on the report.

A matrix is printed out to illustrate the relationships between the names in the two lists and each relationship is designated by an asterisk.

A summary is then produced by counting the number of asterisks appearing in each row and each column of the matrix.

Usages

When a list of ELEMENT and/or GROUP names are given as input to the command producing the report and the CONTAINED parameter is specified, the report aids the analyst by identifying which GROUP and ELEMENT names are not incorporated into higher level information structures. It aids the physical system designer by determining utilization of ELEMENT and GROUP names by the logical information structures within the target system description.

Generating the report (using the CONSISTS parameter) for SET, INPUT, OUTPUT, ENTITY and GROUP names determines which of these have identical structures, with respect to one another, and which are empty, i.e., have no CONSISTS relationships. This also identifies similarities in the structures for these types of names. The analyst may then use this information to determine if redundant structures have been defined, if the description is incomplete, etc.

Example

Figure 21 presents the CONSISTS MATRIX REPORT generated with the CONSISTS parameter in effect and using all INPUT and OUTPUT names in a particular data base as input.

Figure 22 presents the report generated with the CONTAINED parameter in effect and using all GROUP names in a particular data base as input. The Analyzer commands used to generate this example were:

```
NAME-GEN S='GROUP'
CONSISTS-MATRIX CONTAINED
```

FIGURE_21

CONSISTS MATRIX REPORT

PARAMETERS FOR: CM

FILE CONSISTS

URA315:CONCOL : THE FOLLOWING DO NOT CONSIST OF ANYTHING:
employee-information
paysystem-outputs

	ROW NAMES	
1	employee-name	GROUP
2	social-security-number	ELEMENT
3	termination-date	ELEMENT
4	employee-identification-number	ELEMENT
5	employment-status	ELEMENT
6	personal-data	GROUP
7	hourly-job-data	GROUP
8	salaried-job-data	GROUP
9	address	GROUP
10	number-of-deductions	ELEMENT
11	current-date	ELEMENT
12	status-code	ELEMENT
13	pay-date	ELEMENT
14	regular-hours-worked	ELEMENT
15	overtime-hours-worked	ELEMENT
16	hours-per-day	ELEMENT
17	error-listing-entry	GROUP
18	error-code	ELEMENT
19	hired-report-entry	GROUP
20	h-emp-report-entry	GROUP
21	pay-stub	GROUP
22	check	GROUP
23	s-emp-report-entry	GROUP
24	term-report-entry	GROUP

	COLUMN NAMES	
1	employee-information	INPUT
2	employment-termination-form	INPUT
3	hourly-employment-form	INPUT
4	salaried-employment-form	INPUT
5	tax-withholding-certificate	INPUT
6	time-card	INPUT
7	error-listing	OUTPUT
8	hired-employee-report	OUTPUT
9	hourly-employee-report	OUTPUT
10	pay-statement	OUTPUT
11	paysystem-outputs	OUTPUT
12	salaried-employee-report	OUTPUT
13	terminated-employee-report	OUTPUT

THE ROWS ARE CONTAINED IN THE COLUMNS WITH *S

FIGURE_21

CONSISTS MATRIX REPORT

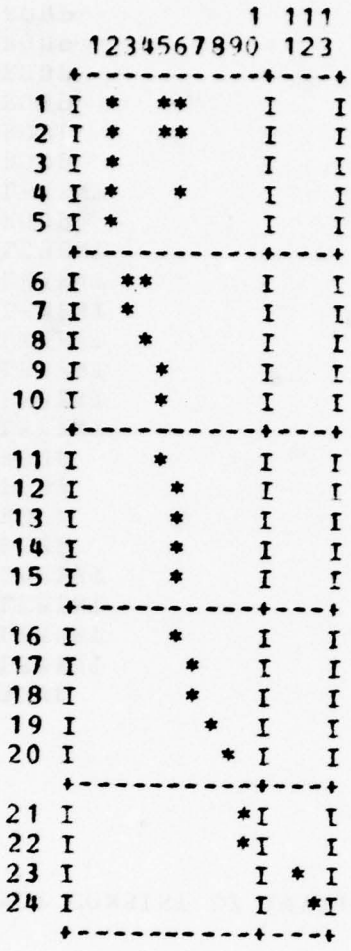


FIGURE_21

CONSISTS MATRIX REPORT

THE NUMBER OF COLUMNS THAT CONTAIN THE ROWS

	ROW	TYPE	COUNT
1	employee-name	GROUP	3
2	social-security-number	ELEMENT	3
4	employee-identification-number	ELEMENT	2
6	personal-data	GROUP	2
3	termination-date	ELEMENT	1
5	employment-status	ELEMENT	1
7	hourly-job-data	GROUP	1
8	salaried-job-data	GROUP	1
9	address	GROUP	1
10	number-of-deductions	ELEMENT	1
11	current-date	ELEMENT	1
12	status-code	ELEMENT	1
13	pay-date	ELEMENT	1
14	regular-hours-worked	ELEMENT	1
15	overtime-hours-worked	ELEMENT	1
16	hours-per-day	ELEMENT	1
17	error-listing-entry	GROUP	1
18	error-code	ELEMENT	1
19	hired-report-entry	GROUP	1
20	h-emp-report-entry	GROUP	1
21	pay-stub	GROUP	1
22	check	GROUP	1
23	s-emp-report-entry	GROUP	1
24	term-report-entry	GROUP	1

THE NUMBER OF ROWS CONTAINED IN THE COLUMNS

	COLUMN	TYPE	COUNT
6	time-card	INPUT	8
2	employment-termination-form	INPUT	5
5	tax-withholding-certificate	INPUT	5
3	hourly-employment-form	INPUT	2
4	salaried-employment-form	INPUT	2
7	error-listing	OUTPUT	2
10	pay-statement	OUTPUT	2

FIGURE_21

CONSISTS MATRIX REPORT

8	hired-employee-report	OUTPUT	1
9	hourly-employee-report	OUTPUT	1
12	salaried-employee-report	OUTPUT	1
13	terminated-employee-report	OUTPUT	1
1	employee-information	INPUT	0
11	paysystem-outputs	OUTPUT	0

FIGURE_22

CONSISTS MATRIX REPORT

PARAMETERS FOR: CM

FILE CONTAINED

URA311:CONROW : THE FOLLOWING ARE NOT CONTAINED IN ANYTHING:
department-update-data
emp-termination-data
h-derived-pay-data
hourly-emp-pay-data
s-derived-pay-data
salaried-emp-pay-data
time-card-data

	ROW NAMES		COLUMN NAMES	
1	address	GROUP	personal-data	GROUP
2	check	GROUP	tax-withholding-certificate	INPUT
3	department-update-data	GROUP	term-report-entry	GROUP
4	emp-termination-data	GROUP	hourly-employee-information	ENTITY
5	employee-name	GROUP	salaried-employee-information	ENTITY
6	error-listing-entry	GROUP	pay-statement	OUTPUT
7	h-derived-pay-data	GROUP	time-card	INPUT
8	h-emp-report-entry	GROUP	employment-termination-form	INPUT
9	hired-report-entry	GROUP	error-listing-entry	GROUP
10	hourly-emp-pay-data	GROUP	h-emp-report-entry	GROUP
11	hourly-job-data	GROUP	s-emp-report-entry	GROUP
12	pay-stub	GROUP	check	GROUP
13	personal-data	GROUP	pay-stub	GROUP
14	s-derived-pay-data	GROUP	hired-report-entry	GROUP
15	s-emp-report-entry	GROUP	error-listing	OUTPUT
16	salaried-emp-pay-data	GROUP	hourly-employee-report	OUTPUT
17	salaried-job-data	GROUP	hired-employee-report	OUTPUT
18	term-report-entry	GROUP	hourly-employment-form	INPUT
19	time-card-data	GROUP	salaried-employment-form	INPUT
			20 salaried-employee-report	OUTPUT
			21 terminated-employee-report	OUTPUT

THE ROWS ARE CONTAINED IN THE COLUMNS WITH *S

FIGURE_22

CONSISTS MATRIX REPORT

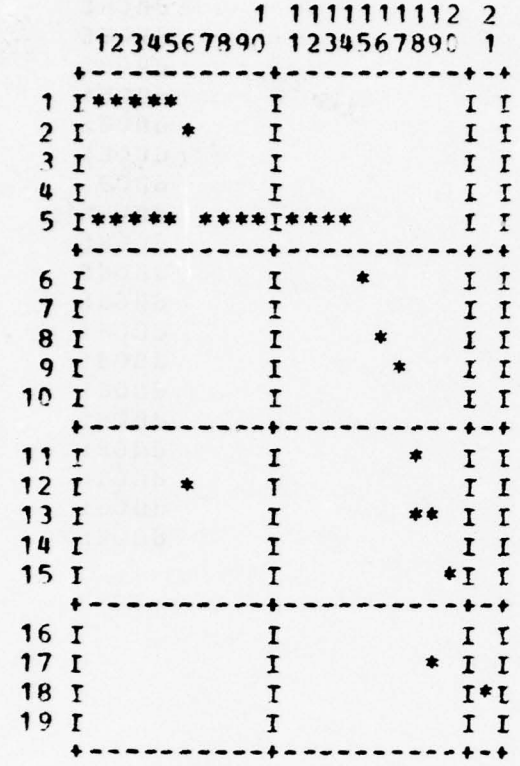


FIGURE_22

CONSISTS MATRIX REPORT

THE NUMBER OF COLUMNS THAT CONTAIN THE ROWS

	ROW	TYPE	COUNT
5	employee-name	GROUP	13
1	address	GROUP	5
13	personal-data	GROUP	2
2	check	GROUP	1
6	error-listing-entry	GROUP	1
8	h-emp-report-entry	GROUP	1
9	hired-report-entry	GROUP	1
11	hourly-job-data	GROUP	1
12	pay-stub	GROUP	1
15	s-emp-report-entry	GROUP	1
17	salaried-job-data	GROUP	1
18	term-report-entry	GROUP	1
3	department-update-data	GROUP	0
4	emp-termination-data	GROUP	0
7	h-derived-pay-data	GROUP	0
10	hourly-emp-pay-data	GROUP	0
14	s-derived-pay-data	GROUP	0
16	salaried-emp-pay-data	GROUP	0
19	time-card-data	GROUP	0

THE NUMBER OF ROWS CONTAINED IN THE COLUMNS

	COLUMN	TYPE	COUNT
1	personal-data	GROUP	2
2	tax-withholding-certificate	INPUT	2
3	term-report-entry	GROUP	2
4	hourly-employee-information	ENTITY	2
5	salaried-employee-information	ENTITY	2
6	pay-statement	OUTPUT	2
18	hourly-employment-form	INPUT	2
19	salaried-employment-form	INPUT	2
7	time-card	INPUT	1
8	employment-termination-form	INPUT	1
9	error-listing-entry	GROUP	1
10	h-emp-report-entry	GROUP	1

FIGURE_22

CONSISTS MATRIX REPORT

11	s-emp-report-entry	GROUP	1
12	check	GROUP	1
13	pay-stub	GROUP	1
14	hired-report-entry	GROUP	1
15	error-listing	OUTPUT	1
16	hourly-employee-report	OUTPUT	1
17	hired-employee-report	OUTPUT	1
20	salaried-employee-report	OUTPUT	1
21	terminated-employee-report	OUTPUT	1

CONTENTS REPORT

Purpose

To allow the user to view entire data structures (all levels) described in the Analyzer data base as implied by the use of the CONSISTS statement.

Information Presented

The CONTENTS REPORT presents all lower levels of data structures for SET, INPUT, OUTPUT, ENTITY and GROUP names used as input. (Since ELEMENTS may not have CONSISTS information they cannot be used as input to produce the report.) All names which the input names CONSIST of are designated as level 2 names; the names that the level 2 names CONSIST of are designated as level 3 names, etc.

The CONSISTS statement allows network structures to be constructed and so any given name may be CONTAINED in more than one structure, and at different levels in the different structures. The types of names presented in the structures will be INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS.

Format

Each name given as input to the report is identified by a number 1*, 2*, etc., designating its position in the list of input names and also by the number 1 designating it as a level 1 name. All names that are part of its structure are numbered 1 though n according to its position in the structure when printed out and also numbered according to the name's relative level in the structure. Each level 2, 3 and so on is indented to further accent the idea of structure. Each group of names of a given level number are CONTAINED in the preceeding name of the next highest level number. (Level 1 is the highest level number.) For example, the following URL description:

```
ENTITY   hourly-employee-record;
   CONSISTS   employee-name,
              employee-identification-number,
              social-security-number;

GROUP    employee-name;
   CONSISTS   surname,
              initial,
              first-name;
```

would appear as:

```
1*      1 hourly-employee-record
 1       2 employee-name
 2         3 surname
 3         3 initial
 4         3 first-name
 5       2 employee-identification-number
 6       2 social-security-number
```

in the CONTENTS REPORT if the report was generated for hourly-employee-record. If the report was generated for employee-name the following structure would appear:

```
1*      1 employee-name
 1       2 surname
 2       2 initial
 3       2 first-name
```

Options and Alternatives

The user may restrict the number of levels of the data structures presented when a numerical value is assigned to the LEVELS parameter. For example, when LEVELS=2 is given only the names at levels one and two of the data structure are presented in the report. The report normally prints out ALL levels of the data structures.

GROUPS in the structures which do not CONSIST of lower level information or those UNDEFINED names in the structure are flagged by the message:

NO CONSISTS FOR GROUP OR UNDEF

when the NOFLAG parameter is used. An index for the report is produced when the INDEX parameter is used.

Security information concerning the names printed, as specified by the CLASSIFICATION statement, can be printed on the report when the PRINT-SECURITY-INFORMATION parameter is used.

The report may be generated for a single input name (via the NAME parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input is searched for in the data base. If the name is not found, the message:

URA065: MAINCONT: NAME NOT FOUND IN D.B.-

is printed and no structure information is printed for the name.

For each name given as an input, each name which is CONSISTS of is designated as a level 2 name as it is printed out. If this level 2 name CONSISTS of any information, then each name which it CONSISTS of is designated as a level 3 name as it is printed out. This process continues until no more CONSISTS relationships are found or the level specified by the LEVELS parameter is reached.

Names are printed out as they are encountered in the structure.

Usage

For the analyst, the report presents information structures as defined by the use of the CONSISTS statement in a format in which the entire structure can be seen. It is usually most beneficial to generate the CONTENTS REPORT for INPUT, OUTPUT and ENTITY names in the data base since all major information constructs are based around these types of names.

Completeness checks on the structure information in the data base can be performed by specifying the NOFLAG parameter when generating the report. Since all GROUPS should consist of other information (by definition of GROUP) and UNDEFINED names should be resolved, this parameter identifies these incomplete aspects of the structures.

For a complete set of final specifications for a particular target system, the report may be generated to present the logical information structures to be handled by the target. For this purpose it is recommended that the report be generated for all SET names with LEVELS=2 so that the relationships among INPUTS, OUTPUTS and ENTITIES with SETS can be presented, and generated for all INPUT, OUTPUT and ENTITY names so that structures below these names can be viewed. The commands to generate this information are:

```
NAME-GEN        S='SET'
CONTENTS        LEVELS=2

NAME-GEN        S='INPUT OR OUTPUT OR ENTITY' ORDER=BYTYPE
CONTENTS
```

the BYTYPE option is used so that all names of a particular name type are defined together.

When the volume of information which will be presented by the CONTENTS REPORT is unknown, it is a good practice to be conservative in specifying a value for LEVELS rather than allow LEVELS to have the value ALL and risk the possibility of generating dozens of pages of output.

Examples

Figure 23 presents the full data structure for the name hourly-employee-record. This example was produced by the following command:

```
CONTENTS  NAME=hourly-employee-report
```

Figure 24 presents the data structures down to level 2 for all ENTITIES defined in a particular Analyzer data base. The Analyzer commands used to generate this example were:

```
NAME-GEN  S='ENTITY'
 CONTENTS  LEVELS=2
```

FIGURE_23

CONTENTS REPORT

```
PARAMETERS FOR: CONT

   NAME=hourly-employee-report NONCFLAG PRINT-SECURITY-INFORMATION NOINDEX LEVELS=ALL


  1*    1 hourly-employee-report (OUTPUT)
   1     2   h-emp-report-entry (GROUP)
   2       3     employee-name (GROUP)
   3         4       surname (ELEMENT)
   4         4       initial (ELEMENT)
   5         4       first-name (ELEMENT)
   6       3     employee-identification-number (ELEMENT)
                   classified 0
   7       3     department (ELEMENT)
   8       3     gross-pay (ELEMENT)
   9       3     status-code (ELEMENT)
                   classified 0
  10       3     total-hours (ELEMENT)
```

FIGURE_24

CONTENTS REPORT

PARAMETERS FOR: CONT

FILE NONCPLAG PRINT-SECURITY-INFORMATION NOINDEX LEVELS=2

```
1*    1 department-information (ENTITY)
 1      2   department (ELEMENT)
 2      2   supervisor (ELEMENT) (no-of-supervisors)
 3      2   number-of-employees (ELEMENT)
 4      2   total-budget (ELEMENT)
 5      2   remaining-funds (ELEMENT)

2*    1 hourly-employee-information (ENTITY)
          classified 2
          secret 1
          top-secret 2
          limited-security 3
 1      2   employee-name (GROUP)
 2      2   employee-identification-number (ELEMENT)
              classified 0
 3      2   social-security-number (ELEMENT)
              classified 0
 4      2   pay-grade-code (ELEMENT)
              classified 0
 5      2   address (GROUP)
 6      2   phone (ELEMENT)
 7      2   employment-date (ELEMENT)
 8      2   number-of-deductions (ELEMENT)
 9      2   department (ELEMENT)
10      2   cumulative-gross-pay (ELEMENT)
11      2   cumulative-federal-deductions (ELEMENT)
12      2   cumulative-state-deductions (ELEMENT)
13      2   cumulative-fica-deductions (ELEMENT)
14      2   age (ELEMENT)
15      2   sex (ELEMENT)
16      2   status-code (ELEMENT)
              classified 0
17      2   employment-status (ELEMENT)
18      2   cumulative-hours (ELEMENT)

3*    1 salaried-employee-information (ENTITY)
```

FIGURE_24

CONTENTS REPORT

```
 1     2   employee-name (GROUP)
 2     2   employee-identification-number (ELEMENT)
             classified 0
 3     2   social-security-number (ELEMENT)
             classified 0
 4     2   pay-grade-code (ELEMENT)
             classified 0
 5     2   address (GROUP)
 6     2   phone (ELEMENT)
 7     2   employment-date (ELEMENT)
 8     2   number-of-deductions (ELEMENT)
 9     2   department (ELEMENT)
10     2   cumulative-federal-deductions (ELEMENT)
11     2   cumulative-gross-pay (ELEMENT)
12     2   cumulative-state-deductions (ELEMENT)
13     2   cumulative-fica-deductions (ELEMENT)
14     2   age (ELEMENT)
15     2   sex (ELEMENT)
16     2   employment-status (ELEMENT)
17     2   status-code (ELEMENT)
             classified 0
```

DATA PROCESS REPORT

Purpose

This report shows the interaction between information (SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS) defined and the PROCESSES defined for the target system. It also shows the data dependencies among PROCESSES as implied by the Language descriptions of the PROCESSES and possible deficiencies in the descriptions of these PROCESSES.

Information Presented

The information presented in this report is slightly different depending on whether the DATA or PROCESS parameter was used in generating the report.

If the DATA parameter is used in generating the report, the names used as input must be SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and/or ELEMENTS and the report presents all those PROCESSES which manipulate the input names via the RECEIVED, USED, UPDATED, DERIVED and GENERATED statements in a particular Analyzer data base.

If the PROCESS parameter is used in generating the report, the names used as input must be PROCESS names and the report presents all those SET, INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names that each input PROCESS name manipulates via the RECEIVES, USES, UPDATES, DERIVES and GENERATES statements in a particular Analyzer data base.

The interactions among data and PROCESSES are shown as a matrix. An analysis on this matrix presents a summary describing the incomplete aspect of the Language description with respect to the information presented in the matrix. For example, a PROCESS producing information without using any information would be identified in this summary.

A second matrix presents the manner in which the PROCESSES (presented in the first matrix) interact, i.e., how they depend on data produced by other PROCESSES. Again, the information in this matrix is derived from the information in the first matrix.

Finally, an analysis is performed on this matrix and presented in the form of a summary. The summary identifies those PROCESSES with no predecessors (i.e., do not USE or RECEIVE data produced by other PROCESSES) and those with no successors (i.e., do not DERIVE, UPDATE or GENERATE any data used by other PROCESSES).

The two summaries and the second matrix are all produced based on the information presented in the first matrix. Therefore,

items which are designated incomplete in the report may actually be resolved elsewhere in the description in the data base. For example, if the following description was in the data base:

```
PROCESS:        payroll-processing;
 USES:            employee-information;
 DERIVES:            paysystem-outputs;
```

and the DATA PROCESS REPORT was produced for the name employee-information, the PROCESS payroll-processing would be presented since it USES it. The report would then identify payroll-processing as USING data but not UPDATING or DERIVING anything though elsewhere in the description it does. For this reason, it is important to recognize that the comments in the report are made with respect to the information in the first matrix rather than the entire description as it exists in the data base.

Format

Two lists of names are first presented, one labeled ROW NAMES and the other COLUMN NAMES. If the DATA parameter was used in generating the report, the names designated as ROW NAMES are those which were given as input. If the PROCESS parameter was used in generating the report the names designated as COLUMN NAMES are those which were given as input.

In any case, each name under COLUMN NAMES in some way (RECEIVES, USES, etc.) interacts with zero or more names given under ROW NAMES and each name under ROW NAMES is manipulated (USED, DERIVED, etc.) by zero or more, PROCESS names given under COLUMN NAMES.

The DATA PROCESS INTERACTION MATRIX is then printed out to show the relationships between the names designated as ROW NAMES (which are represented by the rows of the matrix) and the names designated as COLUMN NAMES (which are represented by the columns of the matrix). The rows and columns of the matrix are numbered to correspond to the number assigned to each name in the list of ROW NAMES and COLUMN NAMES, respectively.

An entry (R, U, D, A, F, 1 or 2) at the intersection of a particular row and column of the matrix designates that the name represented by the row is manipulated in some way (as defined by the meaning of the entry) by the name represented by the column. A legend is provided as part of the report that defines the meaning of each possible entry. This legend is shown below for purposes of clarification.

(i, j) value	meaning
P	Row i is received or used by column j (input)
U	Row i is updated by column j
D	Row i is derived or generated by column j (output)
A	Row i is input to, updated by, and output of column j (all)
F	Row i is input to and output of column j (flow)
1	Row i is input to and updated by column j
2	Row i is updated by and output of column j

A summary section called the DATA PROCESS INTERACTION MATRIX ANALYSIS is then presented specifying those inconsistencies found in analysis of the DATA PROCESS INTERACTION MATRIX. Inconsistencies for ROW NAMES and COLUMN NAMES are handled separately. Inconsistencies found for ROW NAMES are presented under the DATA heading and are of the following format:

row name (name-type) (row number) inconsistency message

Inconsistencies found for COLUMN NAMES are presented under the PROCESS heading and are of the following format:

column name (row number) inconsistency message

No name-type is necessary because all names under COLUMN NAMES are PROCESSES.

A second matrix, the PROCESS INTERACTION MATRIX, is printed to show relationships implied between PROCESSES in the DATA PROCESS INTERACTION MATRIX. In this matrix both the rows and columns represent those PROCESS names listed under COLUMN NAMES and the rows and columns are numbered to correspond to the appropriate name in COLUMN NAMES.

An asterisk (*) entry at the intersection of a particular row and column of the matrix designates that the PROCESS represented by the row DERIVES or UPDATES some information which is USED by the PROCESS represented by the column.

A summary section called the PROCESS INTERACTION MATRIX ANALYSIS then presents observations on the information in the PROCESS INTERACTION MATRIX. These observations are presented in the following format:

process name (row or column number) observation

An observation is given for each PROCESS which does not use information produced by any of the other PROCESSES, or does not produce information used by any of the other PROCESSES.

Options and Alternatives

The report must be generated using either the DATA or PROCESS parameter. If the DATA parameter is used, all names given as input must be SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. If the PROCESS parameter is used, all names given as input must be PROCESS names.

Various sections of the report can be printed or left out depending on the parameters used when generating it. These parameters and their effects are presented below:

1) DPMAT - the DATA PROCESS INTERACTION MATRIX is included in the report

   NODPMAT - the matrix is not printed

2) DPANL - the DATA PROCESS INTERACTION MATRIX ANALYSIS included in the report

   NODPANL - the analysis is not printed

3) PMAT - the PROCESS INTERACTION MATRIX is included in the report

   NOPMAT - the matrix is not printed

4) PANL - the PROCESS INTERACTION ANALYSIS is included in the report

   NOPANL - the analysis is not printed

The report may be generated for a single input name (via the name parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input is searched for in the data base. If it is not found, the message:

URA143:MAINDP: NAME NOT IN D.B.-

is printed. If the name is found and the DATA parameter is used, the name is checked that it is a SET, INPUT, ENTITY, GROUP or ELEMENT name. If it is not, the message:

URA154:MAINDP: INVALID INPUT NAME TYPE

is printed and the name is not included in the matrix.

If the DATA parameter is used, all PROCESS names which RECEIVE, USE, UPDATE, DERIVE and/or GENERATE each input name are found. The list of all input names and the list of all PROCESS names found are then printed on the report.

If the PROCESS parameter is used, all SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names which interact with each input name are found. The list of all input names and the list of all SET, INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names found are then printed on the report.

The DATA PROCESS INTERACTION MATRIX is then printed with the appropriate value (P, U, D, A, F, 1 or 2) designating a relationship between a column name and row name.

The matrix is then analyzed and inconsistency messages are printed as data diagnostics (for row names representing SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names) and process diagnostics (for column names representing PROCESS names).

These diagnostics are presented below categorized by the name types to which they may apply.

I. <u>DATA DIAGNOSTICS (ROWS)</u>

1) INPUT names

- not RECEIVED by any PROCESS

If no PROCESS names RECEIVE the INPUT name of interest, this diagnostic is printed. If at least one PROCESS RECEIVES the INPUT name, the message is not printed.

- not USED by any PROCESS

If no PROCESS USES the INPUT name of interest, this diagnostic is printed.. If at least one PROCESSES USES the INPUT name, the message is not printed.

2) OUTPUT names

- not GENERATED by any PROCESS

If no PROCESS names GENERATE the OUTPUT name of interest, this diagnostic is printed. If at least one PROCESS GENERATES the OUTPUT name, the message is not printed.

- not DERIVED by any PROCESS

If no PROCESS names DERIVE the OUTPUT name of interest, this diagnostic is printed. If at least one PROCESS DERIVES the OUTPUT name, the message is not printed.

3) ENTITY or SET names

- not DERIVED by any PROCESS

If no PROCESS names DERIVE the ENTITY or SET name of interest, this diagnostic is printed. If at least one PROCESS DERIVES the name, the message is not printed.

- DERIVED but not USED by any PROCESS

If at least one PROCESS name DERIVES and no PROCESS names USE the ENTITY or SET name of interest, then this diagnostic is printed. There are 3 conditions that will cause this message not to be printed.

i) the name is not DERIVED by any PROCESS.

ii) the name is USED by at least one PROCESS.

iii) both i and ii.

- UPDATED but not USED by any PROCESS

If at least one PROCESS UPDATES and no PROCESS USES the ENTITY or SET name of interest, this diagnostic is printed. There are 3 conditions that will cause this message not to be printed:

i) the name is not UPDATED by any PROCESS.

ii) the name is USED by at least one PROCESS.

iii) both i and ii.

4) GROUP or ELEMENT

- not DERIVED, UPDATED, or USED by any PROCESS

If no PROCESS names DERIVE, UPDATE or USE the GROUP or ELEMENT name of interest, this diagnostic is printed. There are several conditions that will cause this message not to be printed:

i) at least one PROCESS DERIVES the name.

ii) at least one PROCESS UPDATES the name.

iii) at least one PROCESS USES the name.

iv) all 3 or any combination of the above.

NOTE: it is only necessary that one of the first 3 conditions is satisfied for the message not to be printed.

II. PROCESS DIAGNOSTICS (COLUMNS)

- does not interact with any data

If the PROCESS of interest does not interact with any SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT names, this diagnostic is printed. If at least one name interacts with this PROCESS, this message is not printed.

- USES data, but does not DERIVE or UPDATE anything

If the PROCESS of interest USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT name and does not DERIVE or UPDATE any, this diagnositc is printed. There are several conditions where this message will not be printed:

i) the PROCESS does not USE any SET, INPUT, ENTITY, GROUP or ELEMENT.

ii) the PROCESS DERIVES at least one SET, OUTPUT, ENTITY, GROUP or ELEMENT.

iii) the PROCESS UPDATES at least one SET, ENTITY, GROUP or ELEMENT.

iv) all 3 or any combination of the above.

- DERIVES something but does not USE anything

If the PROCESS of interest DERIVES at least one SET, ENTITY, GROUP or ELEMENT and does not USE any, this diagnostic is printed. There are 3 conditions where this message will not be printed:

i) the PROCESS does not DERIVE any SET, OUTPUT, ENTITY, GROUP or ELEMENT.

ii) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT.

iii) both i and ii.

- UPDATES something but does not USE anything

If the PROCESS of interest UPDATES at least one SET, ENTITY, GROUP or ELEMENT and does not USE any, this diagnostic is printed. There are 3 conditions where this message will not be printed:

i) the PROCESS does not UPDATE any SET, ENTITY, GROUP or ELEMENT.

ii) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or

ELEMENT.

iii) both i and ii.

The PROCESS INTERACTION MATRIX is then produced using the data in the first matrix. The entries in a given column (representing a PROCESS name) of the DATA PROCESS INTERACTION MATRIX are compared with the entries for each other column. If an entry in the given column designates that the PROCESS USES information which is DERIVED or UPDATED by the column being compared with, an * entry is made in the PROCESS INTERACTION matrix at the column or row, respectively, representing these two PROCESS names. The matrix is then analyzed and observations are produced from this analysis. These observations are presented below.

- no interaction with other PROCESSES

If a PROCESS does not USE an object and also does not DERIVE or UPDATE an object, this diagnostic is printed. There are 3 conditions that will cause this message not to be printed:

i) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT.

ii) the PROCESS UPDATES or DERIVES a SET, OUTPUT, ENTITY, GROUP or ELEMENT.

iii) both i and ii.

- no predecessors for this PROCESS

If the PROCESS of interest does not USE any objects but DERIVES or UPDATES at lease one object, this diagnostic is printed. The 3 conditions that will cause this message not to be printed are:

i) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT.

ii) the PROCESS does not DERIVE or UPDATE any SETS, OUTPUTS, ENTITIES, GROUPS or ELEMENTS.

iii) both i and ii.

- no successors for this PROCESS

If the PROCESS of interest USES at least one object but does not DERIVE or UPDATE any object, this diagnostic is printed. The 3 conditions that will cause this message not to be printed are:

i) the PROCESS does not USE any SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS.

ii) the PROCESS DERIVES or UPDATES at least one SET, OUTPUT,

ENTITY, GROUP or ELEMENT.

iii) both i and ii.

Usage

When the report is generated using the DATA parameter, it aids in presenting the utilization of SET, INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names defined. This aids in identifying which names are not being utilized. For those names which are being utilized, it presents all the PROCESS names which utilize them.

When the report is generated using the PROCESS parameter, it presents all the data required for each particular PROCESS. It also aids in identifying PROCESS names which do not interact with data or are not consistently defined with respect to the manner in which they use data.

The PROCESS INTERACTION MATRIX may be used by designers to plan out the logic of the target system because it presents the data dependencies among the PROCESSES defined.

The following completeness checks can be made for a target system description based on information presented in the report:

- ALL INPUTS RECEIVED by some PROCESS
- ALL INPUTS USED by some PROCESS
- ALL OUTPUTS GENERATED by some PROCESS
- ALL OUTPUTS DERIVED by some PROCESS
- ALL ENTITIES and SETS DERIVED by some PROCESS
- ALL ENTITIES and SETS DERIVED and USED by some PROCESS
- ALL ENTITIES and SETS are UPDATED and USED by some PROCESS
- ALL GROUPS and ELEMENTS are DERIVED or UPDATED or USED by some PROCESS
- ALL PROCESSES USE data and DERIVE or UPDATE data
- ALL PROCESSES which DERIVE data also USE data
- ALL PROCESSES which UPDATE data also USE data
- ALL PROCESSES interact with data in some way

The report may also be used to aid in determining if the description of the target system was specified consistently with respect to the use of language statements. In particular, it

determines:

- whether or not the use of RECEIVES and GENERATES statements in describing the system flow aspect of the system is consistent.

- whether or not the use of USES, UPDATES, and DERIVES statement in describing the data derivation aspect of the system is consistent.

Examples

Figure 25 presents the DATA PROCESS REPORT generated for all INPUT, OUTPUT and ENTITY names defined for a particular target system description. This example was produced using the Analyzer commands:

```
NAME-GEN S='INPUT OR OUTPUT OR ENTITY'
DATA-PROCESS DATA
```

Figure 26 presents the report generated for low level PROCESSES defined in the description. These PROCESSES were identified by the KEYWORD "terminal" and the report was produced by the following Analyzer commands:

```
NAME-GEN S='PROCESS AND KEY=terminal'
DATA-PROCESS PROCESS
```

FIGURE_25

DATA PROCESS REPORT

PARAMETERS FOR: DP

FILE DATA DPMAT DPANL PMAT PANL

THE ROWS ARE DATA NAMES, THE COLUMNS ARE PROCESS NAMES.

	ROW NAMES	
1	department-information	ENTITY
2	hourly-employee-information	ENTITY
3	salaried-employee-information	ENTITY
4	employee-information	INPUT
5	employment-termination-form	INPUT
6	hourly-employment-form	INPUT
7	salaried-employment-form	INPUT
8	tax-withholding-certificate	INPUT
9	time-card	INPUT
10	error-listing	OUTPUT
11	hired-employee-report	OUTPUT
12	hourly-employee-report	OUTPUT
13	pay-statement	OUTPUT
14	paysystem-outputs	OUTPUT
15	salaried-employee-report	OUTPUT
16	terminated-employee-report	OUTPUT

	COLUMN NAMES	
1	hourly-information-creation	PROCESS
2	term-report-entry-generation	PROCESS
3	hourly-employee-processing	PROCESS
4	salaried-information-creation	PROCESS
5	salaried-employee-processing	PROCESS
6	payroll-processing	PROCESS
7	terminating-emp-processing	PROCESS
8	hire-report-entry-generation	PROCESS
9	new-employee-processing	PROCESS

FIGURE_25

DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX

(i,j) value	meaning
R	Row i is received or used by column j (input)
U	Row i is updated by column j
D	Row i is derived or generated by column j (output)
A	Row i is input to, updated by, and output of column j (all)
P	Row i is input to and output of column j (flow)
1	Row i is input to and updated by column j
2	Row i is updated by and output of column j

	1	2	3	4	5	6	7	8	9
1	I				I				I
2	I	D	R	R	I				I
3	I		R		D	RI			I
4	I				I	R			I
5	I				I		R		I
6	I	R			I			R	RI
7	I			R	I			R	RI
8	I				I				RI
9	I		R		I				I
10	I		D		DI				I
11	I				I				DI
12	I		D		I				I
13	I		D		DI				I
14	I				I	D			I
15	I				DI				I
16	I				I		D		I

FIGURE_25

DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX ANALYSIS

DATA
----
department-information (ENTITY) (ROW 1) NOT DERIVED BY ANY PROCESS

PROCESSES
----
term-report-entry-generation (COLUMN 2) USES DATA, BUT DOES NOT DERIVE OR UPDATE ANYTI
hire-report-entry-generation (COLUMN 3) USES DATA, BUT DOES NOT DERIVE OR UPDATE ANYTH

FIGURE_25

DATA PROCESS REPORT

PROCESS INTERACTION MATRIX (INCIDENCE)

The rows and columns are process names from above.
An asterisk in (i,j) means that something derived
or updated by process i is used by process j.

	1	2	3	4	5	6	7	8	9
1	*	*							
2									
3									
4	*		*						
5									
6									
7									
8									
9									

FIGURE_25

DATA PROCESS REPORT

PROCESS INTERACTION MATRIX ANALYSIS

hourly-information-creation	(ROW/COL 1)	NO PREDECESSORS FOR THIS PROCESS
term-report-entry-generation	(ROW/COL 2)	NO SUCCESSORS FOR THIS PROCESS
hourly-employee-processing	(ROW/COL 3)	NO SUCCESSORS FOR THIS PROCESS
salaried-information-creation	(ROW/COL 4)	NO PREDECESSORS FOR THIS PROCESS
salaried-employee-processing	(ROW/COL 5)	NO SUCCESSORS FOR THIS PROCESS
payroll-processing	(ROW/COL 6)	NO INTERACTION, BUT HAS SUBPARTS
terminating-emp-processing	(ROW/COL 7)	NO INTERACTION, BUT HAS SUBPARTS AND IS PART
hire-report-entry-generation	(ROW/COL 8)	NO INTERACTION, BUT IS PART OF ANOTHER PROCES
new-employee-processing	(ROW/COL 9)	NO INTERACTION, BUT HAS SUBPARTS AND IS PART

FIGURE_26

DATA PROCESS REPORT

PARAMETERS FOR: DP

FILE PROCESS DPMAT DPANL PMAT PANL

THE ROWS ARE DATA NAMES, THE COLUMNS ARE PROCESS NAMES.

	ROW NAMES	
1	cumulative-federal-deductions	ELEMENT
2	federal-tax	ELEMENT
3	cumulative-fica-deductions	ELEMENT
4	fica-tax	ELEMENT
5	remaining-funds	ELEMENT
6	department	ELEMENT
7	gross-pay	ELEMENT
8	cumulative-gross-pay	ELEMENT
9	pay-grade-code	ELEMENT
10	total-hours	ELEMENT
11	cumulative-hours	ELEMENT
12	net-pay	ELEMENT
13	total-deductions	ELEMENT
14	error-code	ELEMENT
15	employee-identification-number	ELEMENT
16	social-security-number	ELEMENT
17	status-code	ELEMENT
18	cumulative-state-deductions	ELEMENT
19	state-tax	ELEMENT
20	number-of-deductions	ELEMENT
21	error-messages	*** UNDEFINED ***
22	error-flags	*** UNDEFINED ***
23	pay-date	ELEMENT
24	regular-hours-worked	ELEMENT
25	overtime-hours-worked	ELEMENT

	COLUMN NAMES	
1	federal-deductions-update	PROCESS
2	fica-dedictions-update	PROCESS
3	funds-uplate	PROCESS
4	gross-pay-update	PROCESS
5	h-gross-pay-computation	PROCESS
6	hours-uplate	PROCESS
7	net-pay-computation	PROCESS
8	pay-compitation-validation	PROCESS
9	s-gross-pay-computation	PROCESS
10	state-deductions-update	PROCESS
11	tax-computation	PROCESS
12	time-card-validation	PROCESS
13	total-deductions-computation	PROCESS
14	total-hours-computation	PROCESS

FIGURE_26

DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX

(i,j) value	meaning
R	Row i is received or used by column j (input)
U	Row i is updated by column j
D	Row i is derived or generated by column j (output)
A	Row i is input to, updated by, and output of column j (all)
F	Row i is input to and output of column j (flow)
1	Row i is input to and updated by column j
2	Row i is updated by and output of column j

FIGURE_26

DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX ANALYSIS

FIGURE_26

DATA PROCESS REPORT

PROCESS INTERACTION MATRIX (INCIDENCE)

The rows and columns are process names from above.
An asterisk in (i,j) means that something derived
or updated by process i is used by process j.

	1 2 3 4 5 6 7 8 9	10	11 12 13 14
1			
2			
3			
4			
5	** *	*	
6			
7			
8			
9	** *	*	
10			
11	**	*	*
12			
13	*		
14	**		

FIGURE_26

DATA PROCESS REPORT

PROCESS INTERACTION MATRIX ANALYSIS

federal-deductions-update	(ROW/COL	1)	NO SUCCESSORS FOR THIS PROCESS
fica-deductions-update	(ROW/COL	2)	NO SUCCESSORS FOR THIS PROCESS
funds-update	(ROW/COL	3)	NO SUCCESSORS FOR THIS PROCESS
gross-pay-update	(ROW/COL	4)	NO SUCCESSORS FOR THIS PROCESS
hours-update	(ROW/COL	6)	NO SUCCESSORS FOR THIS PROCESS
net-pay-computation	(ROW/COL	7)	NO SUCCESSORS FOR THIS PROCESS
pay-computation-validation	(ROW/COL	8)	NO INTERACTION, BUT IS PART OF ANOTHER PROCESS
s-gross-pay-computation	(ROW/COL	9)	NO PREDECESSORS FOR THIS PROCESS
state-deductions-update	(ROW/COL	10)	NO SUCCESSORS FOR THIS PROCESS
time-card-validation	(ROW/COL	12)	NO INTERACTION, BUT IS PART OF ANOTHER PROCESS
total-hours-computation	(ROW/COL	14)	NO PREDECESSORS FOR THIS PROCESS

DICTIONARY REPORT

Purpose

This report presents definitions attached to names used in a Language description and is intended as an aid in communication among persons interested in the description.

Information Presented

This report prints out the following information about each name used as input when generating the report and the appropriate parameters are used:

- Name type of the name
- DESCRIPTION comment entry for the name
- SYNONYMS associated with the name
- RESPONSIBLE PROBLEM DEFINER for the name's description
- KEYWORDS associated with the name

All of the above information is readily available from the contents of the data base.

Format

A dictionary entry is presented for each name given as input to the software producing the report. The first line of each entry consists of:

- A number designating the order the name was read from the input (and consequently presented in the report)
- The name the entry is for
- The name type of the name

The DESCRIPTION, SYNONYM, KEYWORDS and RESPONSIBLE-PROBLEM-DEFINER statements for each name are presented in the following format after the first line of the entry:

```
DESCRIPTION:

        [DESCRIPTION comment entry]

SYNONYMS:     [all SYNONYMS for the name listed two per line]
```

KEYWORDS: [all KEYWORDS for the name listed two per line]

RESP PD: [PROBLEM DEFINER name]

Spacing between dictionary entries may be modified by the NUM-SPACE PARAMETER.

Options and Alternatives

The number of lines skipped between dictionary entries is specified by the NUM-SPACE parameter. By default, 3 lines are skipped but NUM-SPACE may take any value 0 through 10.

The different types of information presented in a report entry can be included in or left out of the report depending on the parameters used when generating it. Each parameter and its effect is presented below:

1) DESCRIPTION	-	the DESCRIPTION comment entry for each name is printed
NODESCRIPTION	-	DESCRIPTION comment entries are not printed
2) KEYWORDS	-	all KEYWORDS associated to each name is printed out
NOKEYWORDS	-	KEYWORDS are not printed
3) RESPONSIBLE-PD	-	the RESPONSIBLE-PROBLEM-DEFINER for each name is printed
NORESPONSIBLE-PD	-	RESPONSIBLE-PROBLEM-DEFINERS are not printed
4) SYNONYMS	-	all SYNONYMS associated to each name are printed out
NOSYNONYMS	-	SYNONYMS are not printed

An INDEX for the report is provided when the INDEX parameter is used.

Analysis

For each name given as input, the software finds the name in the data base. If the name cannot be found, the message:

URA064: MAINDICT: NAME NOT FOUND IN D.B.-

is printed. If the name is found, the information specified by the parameters for the command is retrieved if available for the

name.

Usage

The report is a valuable aid to analysts in maintaining definitions for names in the data base and as a tool for communicating with users of the target system. The DESCRIPTIONS for each name may be okayed or disapproved by the users with respect to what the users require of the target system. As DESCRIPTIONS are modified, the analyst can add, delete or modify other statements to correspond to the DESCRIPTIONS.

If conventions are imposed on the language description required for particular types of names, an effective data dictionary can be formed. For example, by requiring certain KEYWORDS to be assigned to GROUP and ELEMENT names and that each GROUP and ELEMENT name have a DESCRIPTION, the DICTIONARY REPORT would be a good reference for anyone interested in the data described in the target system.

Examples

Figure 27 presents the DICTIONARY REPORT for a single PROCESS name. This example was produced by the command:

```
DICTIONARY NAME=payroll-processing
```

Figure 29 presents the report for several PROCESS names which have the KEYWORD 'independent'. This example was produced by the following Analyzer commands:

```
NAME-GEN S='PROCESS AND KEY=independent'
DICTIONARY
```

FIGURE_27

DICTIONARY REPORT

PARAMETERS FOR: DICT

NAME=payroll-processing NOINDEX DESCRIPTION SYNONYMS KEYWORDS RESPONSIBLE-PD NUM-SPACE=2

1 payroll-processing PROCESS

DESCRIPTION:
This process represents the highest level process in the target system. it accepts and processes all inputs and produces all outputs.

SYNONYMS: payproc p1

RESP PD: michel-j-bastarache

FIGURE_28

DICTIONARY REPORT

PARAMETERS FOR: DICT

FILE NOINDEX DESCRIPTION SYNONYMS KEYWORDS RESPONSIBLE-PD NUM-SPACE=2

1 hourly-employee-processing PROCESS

DESCRIPTION:
This process performs those actions needed to interpret time cards to produce a pay statement for each hourly employee.

KEYWORDS: independent

2 new-employee-processing PROCESS

DESCRIPTION:
This process stores information about new employees and then prints out a corresponding report.

KEYWORDS: independent

3 salaried-employee-processing PROCESS

DESCRIPTION:
This process produces the pay statement for salaried employees once a month.

KEYWORDS: independent

4 terminating-emp-processing PROCESS

DESCRIPTION:
This process deletes data, for those employees who are no longer on the payroll, from the files. It also prints a list of all employees no longer on the payroll.

FIGURE_28

DICTIONARY REPORT

KEYWORDS: independent

EXTENDED PICTURE REPORT

Purpose

To present in graphical format, for each name input, a network of names related to it by structure or by data flow.

Information Presented

Names input to the report may have any of the following types:

ELEMENT
ENTITY
GROUP
INPUT
INTERFACE
OUTPUT
PROCESS
SET

Starting with each input name, one of four pictures will be obtained. These are referred to as structure downward, structure upward, data-flow forward, and data-flow backward.

For each name used as input, the report presents all successors to that name, where successors are found using the relationships listed in the appropriate column of either Table 3 or Table 4. For each of these new names, the report presents all of its successors, finding them in a similar manner. This network continues until the desired number of relationships has been traced, a loop is encountered, or no more relationships are found.

Name Type	Relationships Displayed Structure Downward	Relationships Displayed Structure Upward
ELEMENT		CONTAINED
ENTITY	CONSISTS	CONTAINED
GROUP	CONSISTS	CONTAINED
INPUT	SUBPARTS CONSISTS	PART CONTAINED
INTERFACE	SUBPARTS	PART
OUTPUT	SUBPARTS CONSISTS	PART CONTAINED
PROCESS	SUBPARTS UTILIZES	PART UTILIZED
SET	SUBSETS CONSISTS	SUBSET

TABLE 3
Structure Relationships Displayed in
Extended Picture Report

Name Type	Relationships Displayed Data-Flow Forward	Relationships Displayed Data-Flow Backward
ELEMENT	USED USED TO DERIVE USED TO UPDATE	DERIVED UPDATED
ENTITY	USED USED TO DERIVE USED TO UPDATE	DERIVED UPDATED
GROUP	USED USED TO DERIVE USED TO UPDATE	DERIVED UPDATED
INPUT	RECEIVED USED USED TO DERIVE USED TO UPDATE	GENERATED
INTERFACE	GENERATES	RECEIVES
OUTPUT	RECEIVED	GENERATED DERIVED
PROCESS	GENERATED DERIVES UPDATES	RECEIVES USES USES TO DERIVE USES TO UPDATE
SET	USED USED TO DERIVE USED TO UPDATE	DERIVED UPDATED

Table 4

Data Flow Relationships Displayed in
Extended Picture Report

Format

Each name which appears on the output is shown within a box. The top line of the box indicates the name type, while the bottom line shows the relationship with a name's predecessor. Boxes containing related names are linked by dotted lines.

If a name joins two or more chains (strings of related names) into a loop or loops, every appearance of that name after the first will be linked to a box containing the message, "LOOPS TO PREVIOUS ENTRY."

Output is continued across page boundaries. If the right edge of one page continues to the left edge of a second, the right most column of boxes on the first page will be repeated as the left most column of boxes on the second page, in order to facilitate matching of edges. Similarly, if the bottom edge of one page continues to the top edge of a second, the bottom row of boxes on the first page will be repeated as the top row of boxes on the second page.

Options and Alternatives

The report may be generated for a single name (via the NAME parameter) or for a collection of names, either input by the user or obtained by use of NAME-GEN.

The type of picture to be produced is selected by specifying one of the following parameter pairs:

STRUCTURE DOWNWARD
STRUCTURE UPWARD
DATA-FLOW FORWARD
DATA-FLOW BACKWARD

The number of columns and rows used on the page may be decreased from their default values of 119 and 39, respectively, via the COLUMNS and ROWS parameters. The minimum acceptable values for COLUMNS and ROWS are 38 and 14, respectively.

The number of boxes arranged horizontally or vertically on a page may be decreased from the defaults, which are the maximum numbers that will fit in each direction (depending on COLUMNS and ROWS), in order to make the output less cluttered. The parameters which can be used to do this are HORIZONTAL-BOXES and VERTICAL-BOXES. Their maximum values (for COLUMNS=119 and ROWS=39) are 6. Due to the scheme for continuing pages, their minimum values are 2.

The number of connections to be traced, starting at the given name, may be set at any positive value via the LINKS parameter. The same LINKS value is used for all names input when the FILE parameter is used.

An index, containing each name used on the report and the page(s) on which it appears, may be obtained by specifying the INDEX parameter.

Analysis

Each name given as input is first checked to see that it is in the data base and that it has one of the legal types (ELEMENT, ENTITY, GROUP, INPUT, INTERFACE, OUTPUT, PROCESS, or SET). If the name is not in the data base, the message:

URA370: MAINEP: NAME NOT FOUND IN DATA BASE

will be given. If the name has a type which is not in the list above, the user will receive the message:

URA 371: EPSUCC: NAME NOT ACCEPTABLE TYPE FOR EP REPORT

If the name passes these two tests, it is placed in a data structure which will later be used for output. The name is then used to generate a tree structure as follows. Using the relationships in the appropriate column of Table 1 or Table 2, all successors for the name are retrieved from the data base and placed on a stack. Then, the first name is removed from the stack, placed in its proper location in the data structure, and all successors for that name are retrieved and placed on the stack. This procedure continues, with names being removed from the top of the stack, placed in the data structure, and used to obtain further names, which are then placed on the stack. At any stage of this procedure, no names will be put on the stack if one of the following is true:

1) The current name has no successors.

2) The current name has been encountered earlier, and is therefore at the end of a chain or forms a loop with some portion of a chain traced earlier.

3) The number of links that has been traced on the current chain is equal to the limit set by the LINKS paramter. (For every input name for which this occurs, the message:

URA372: EPSUCC: LINK LIMIT SPECIFIED WAS REACHED

will be printed.)

Thus, in any of these cases the size of the stack will decrease. The entire procedure is complete when, after any search, the stack is empty.

The data structure constructed from all names found as above is broken into page-sized units and is printed a page at a time.

AD-A057 130 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS ANALYZER (URA) USER'S MANUAL H6180/MULTICS/VE--ETC(U)
MAR 77 D TEICHROEW, A HERSHEY, S SPEWAK F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-128 NL

3 OF 6
AD A057130



The process described above is repeated until no more names remain in the input stream.

Usage

The EXTENDED PICTURE is very similar in content to the PICTURE report and therefore, most usages of the PICTURE report apply to the EXTENDED PICTURE report.

In addition, the EXTENDED PICTURE report provides a comprehensive view of the information flow and structure aspects of the target system for inclusion in the final specifications of the system, or as an aid in communicating this information to others.

Problem definers may use the EXTENDED PICTURE report to visually analyze the description of particular objects and the system as a whole, for completeness. Table 5 presents all completeness checks that can be made by visually scanning EXTENDED PICTURE reports.

Examples

Figure 29 presents an EXTENDED PICTURE of structure information for the PROCESS "hourly-employee-processing."

Figure 30 presents and EXTENDED PICTURE of data flow information for the INTERFACE "payroll-department." The amount of information presented was limited by setting LINE=3.

FIGURE_29

EXTENDED PICTURE

PARAMETERS FOR: BP

NAME=hourly-employee-processing NODATA-FLOW STRUCTURE FORWARD NOBACKWARD LINKS=1000 NOINDEX
COLUMNS=110 ROWS=39 HORIZONTAL-BOXES=6 VERTICAL-BOXES=6

FIGURE_29

EXTENDED PICTURE

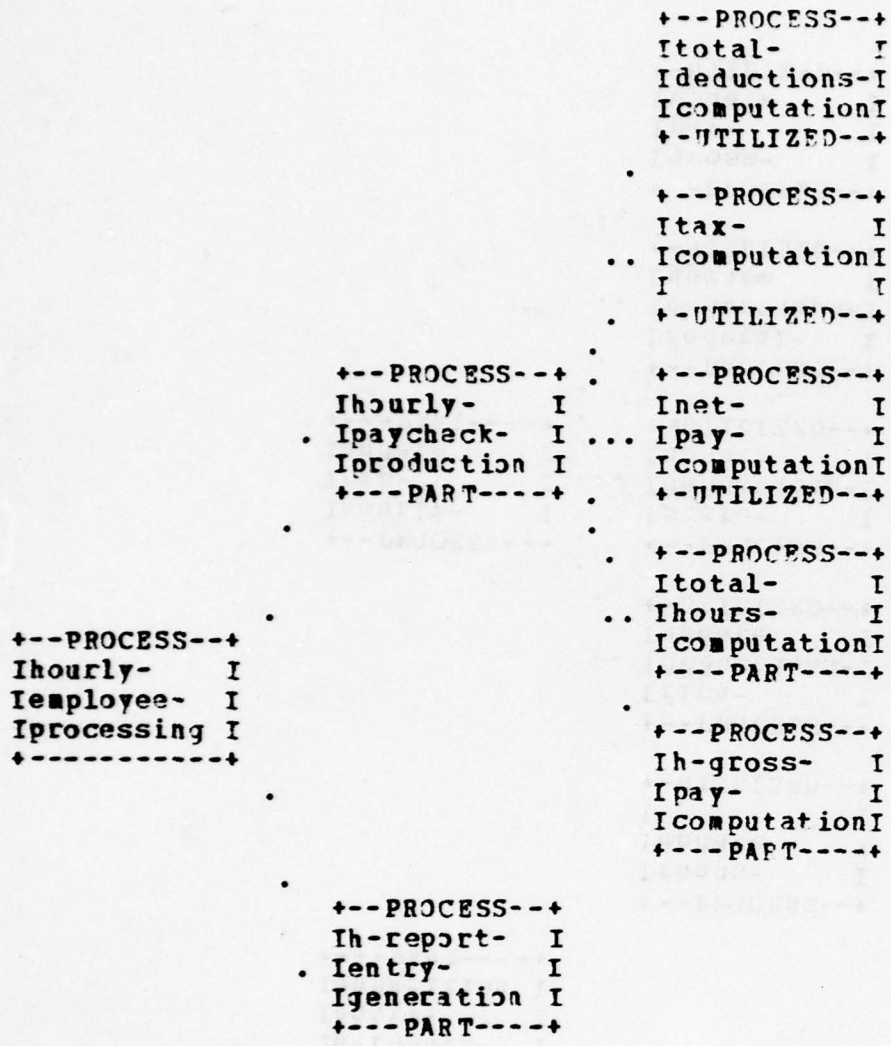


FIGURE_29

EXTENDED PICTURE

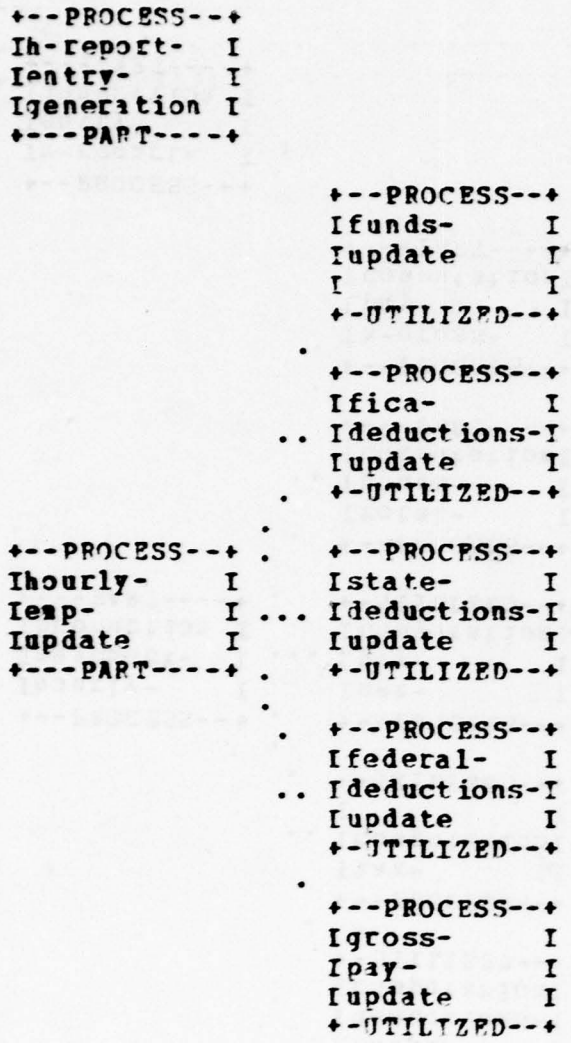


FIGURE_29

EXTENDED PICTURE

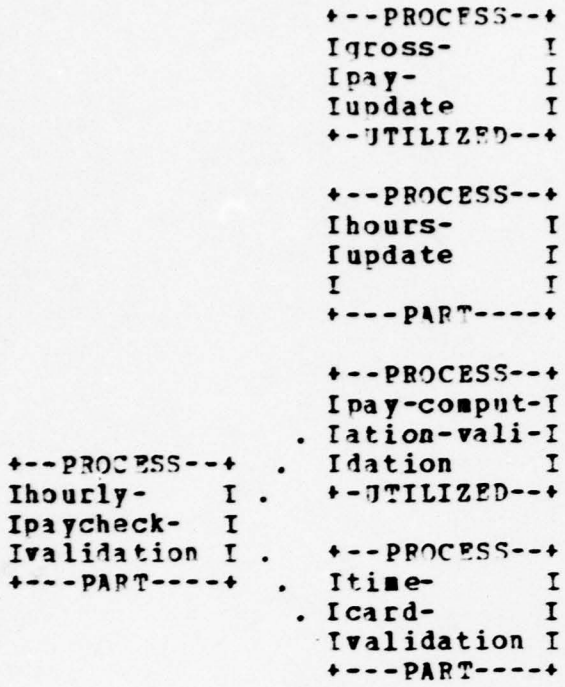


FIGURE_3)

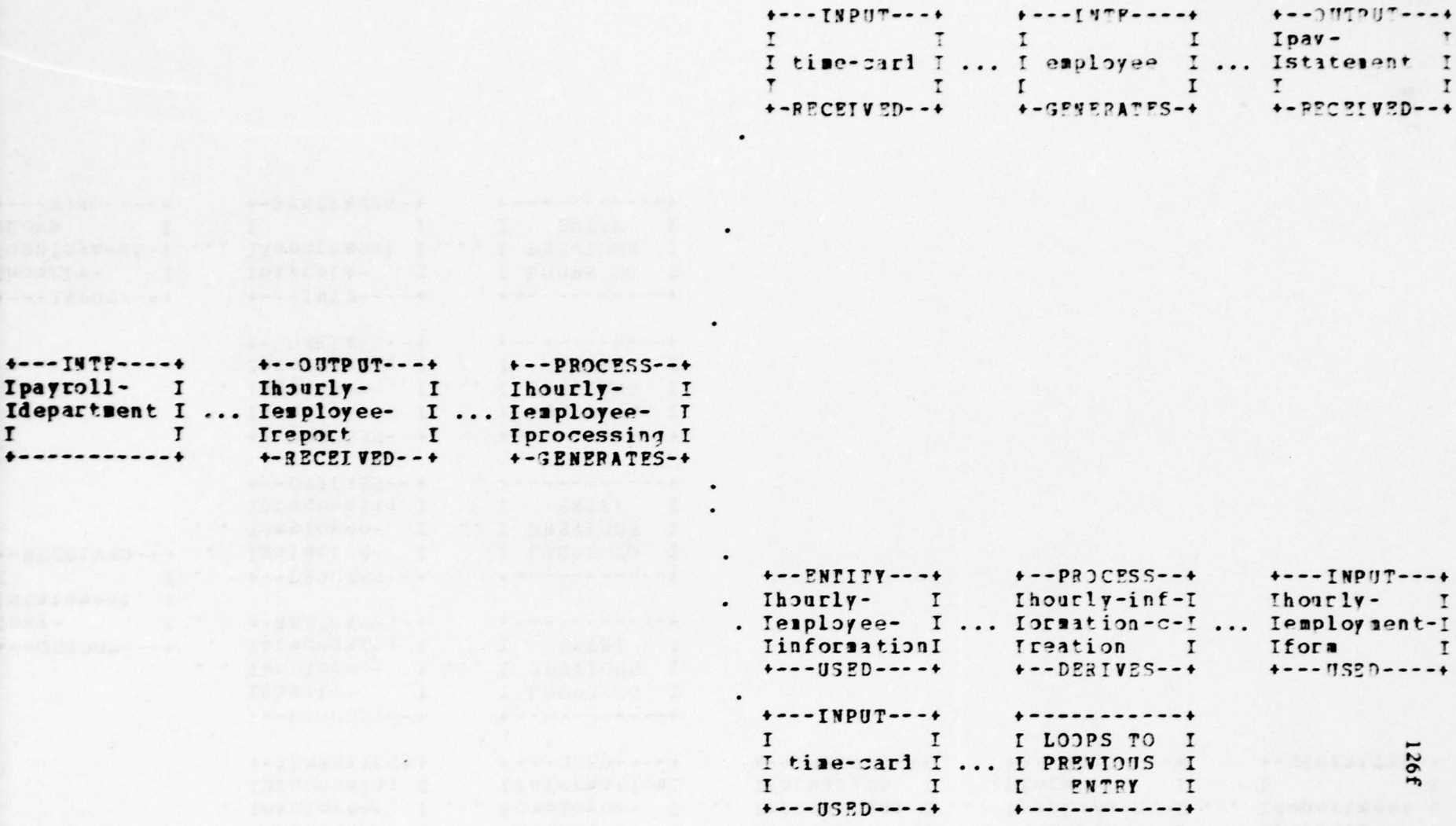
EXTENDED PICTURE

PARAMETERS FOR: EP

NAME=payroll-department DATA-FLOW NOSTRUCTURE NOFORWARD BACKWARD LINKS=1000 NOINDEX
COLUMNS=110 ROWS=39 HORIZONTAL-BOXES=6 VERTICAL-BOXES=6

FIGURE_30

EXTENDED PICTURE



126f

FIGURE_30

EXTENDED PICTURE

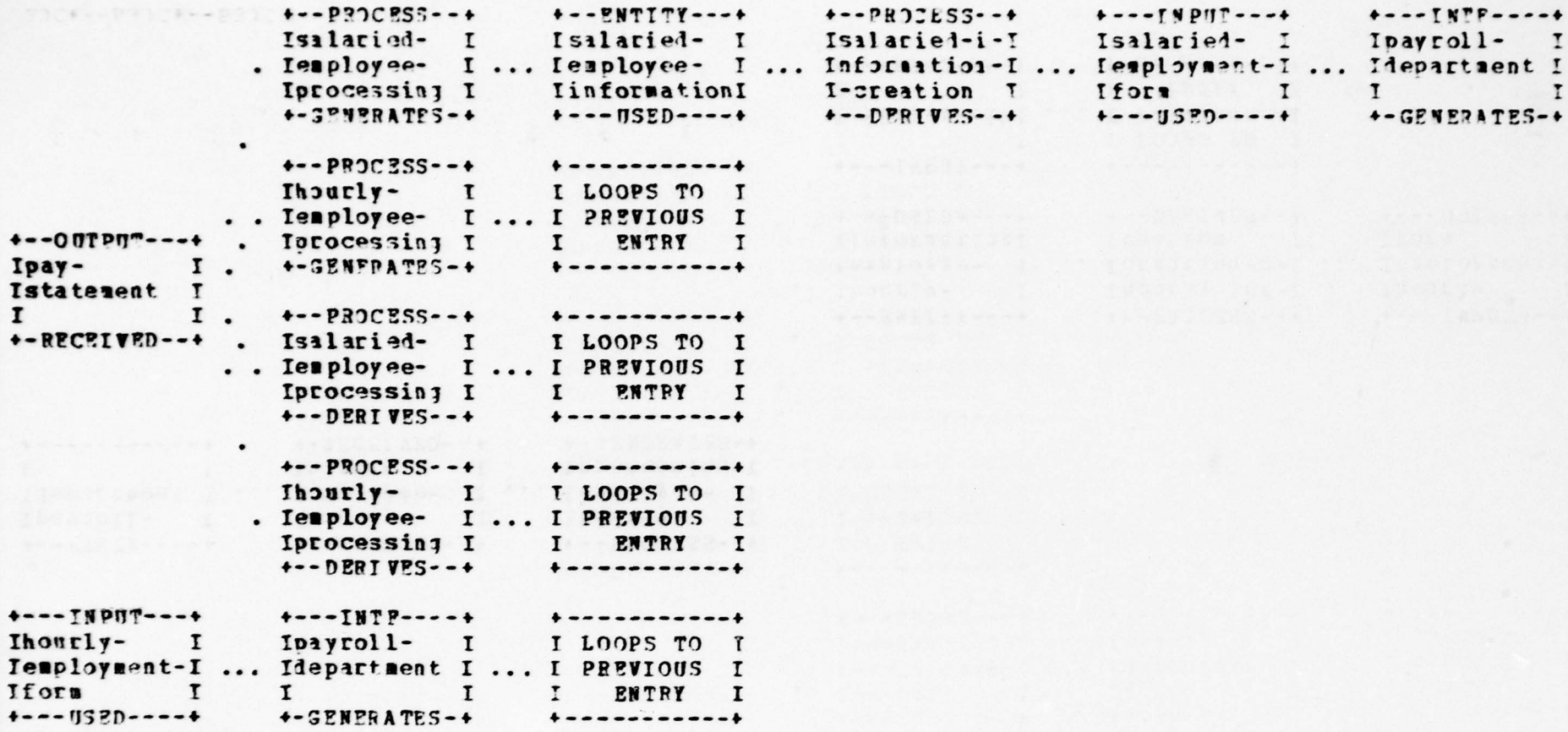


FIGURE_30

EXTENDED PICTURE

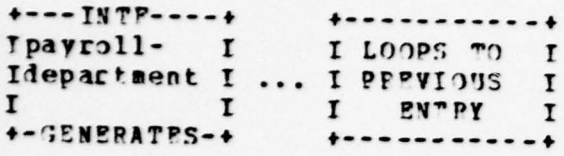


FIGURE_30

EXTENDED PICTURE

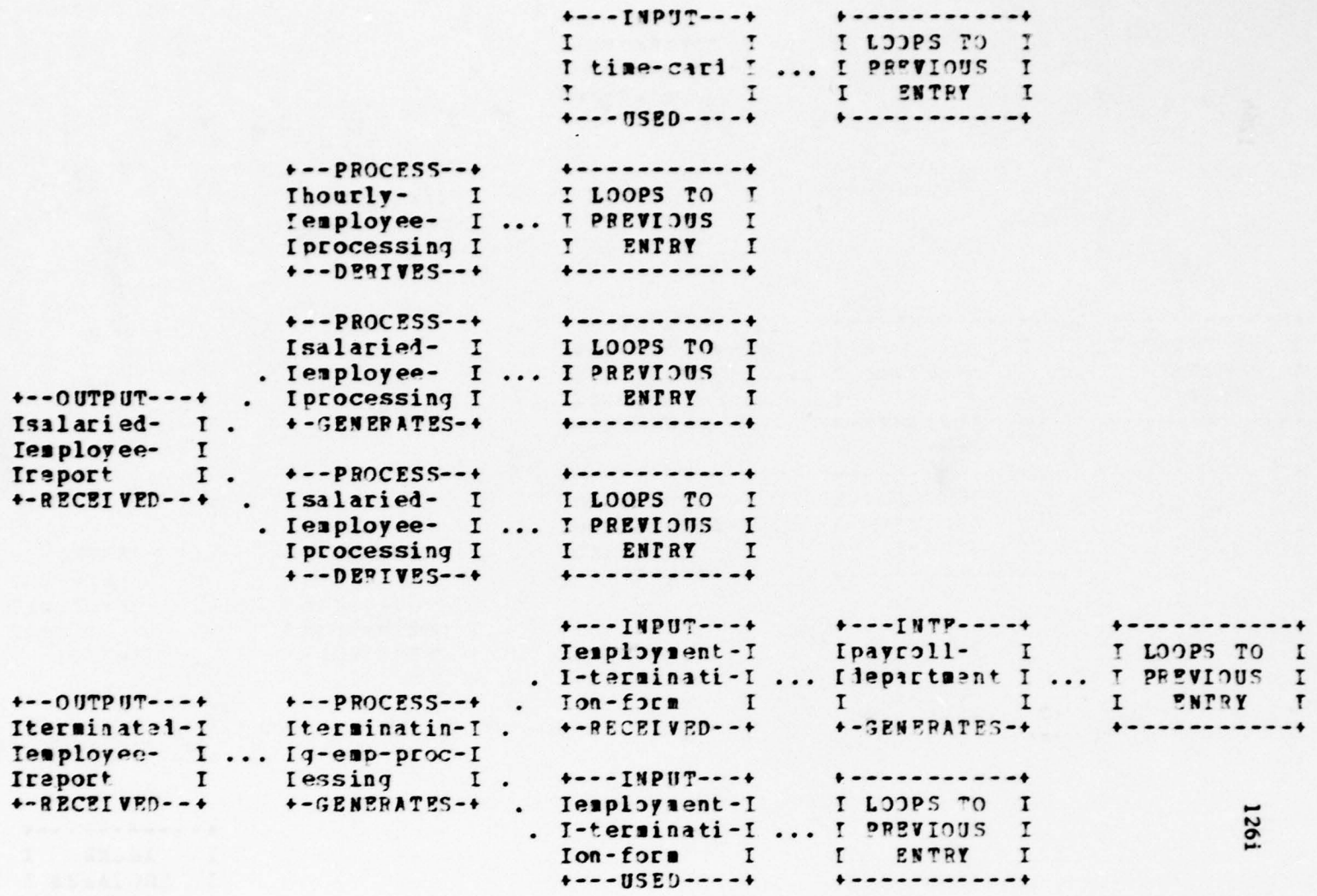


FIGURE_30

EXTENDED PICTURE

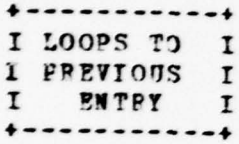
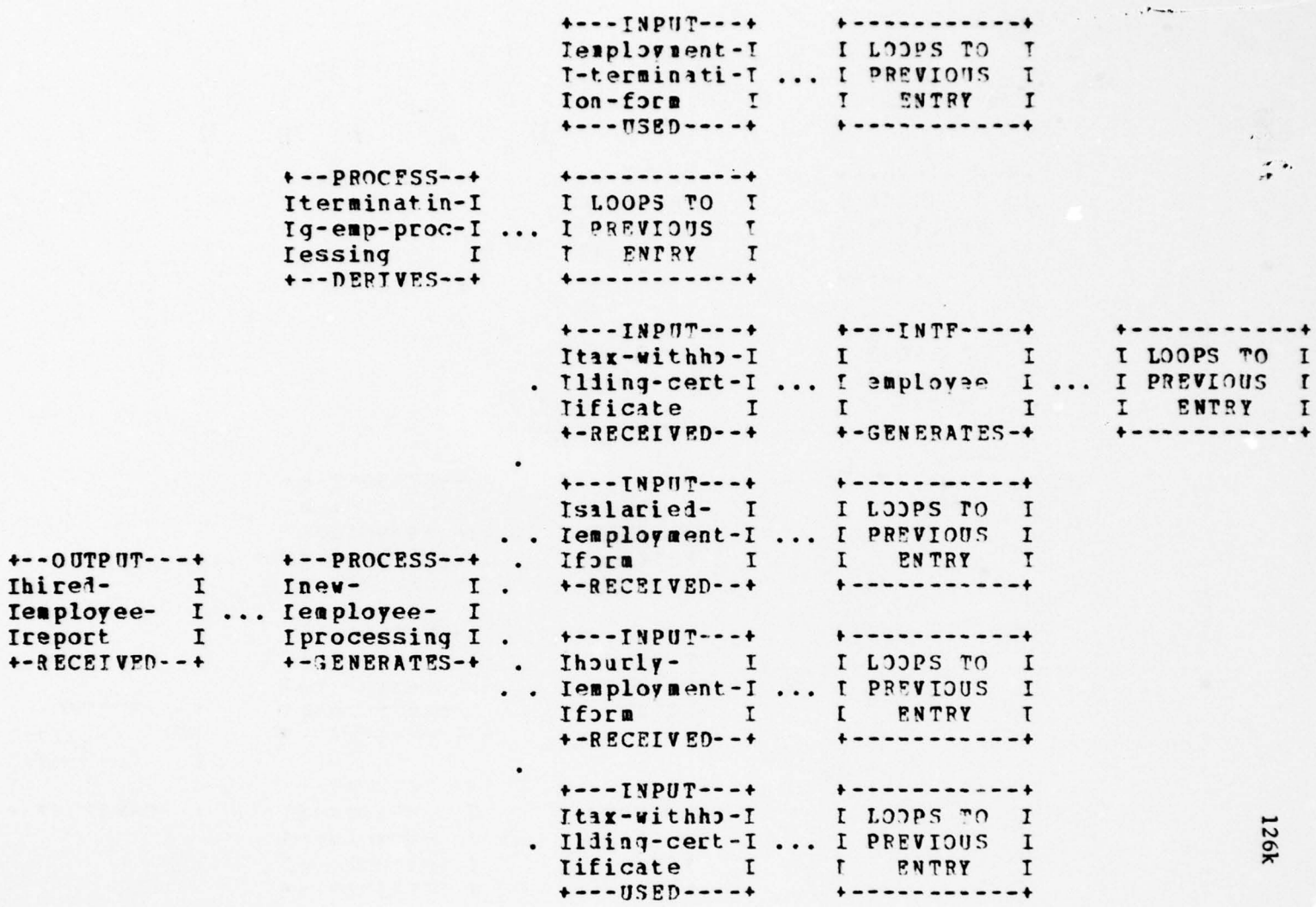


FIGURE_30

EXTENDED PICTURE



126k

FIGURE_30

EXTENDED PICTURE



FIGURE_30

EXTENDED PICTURE

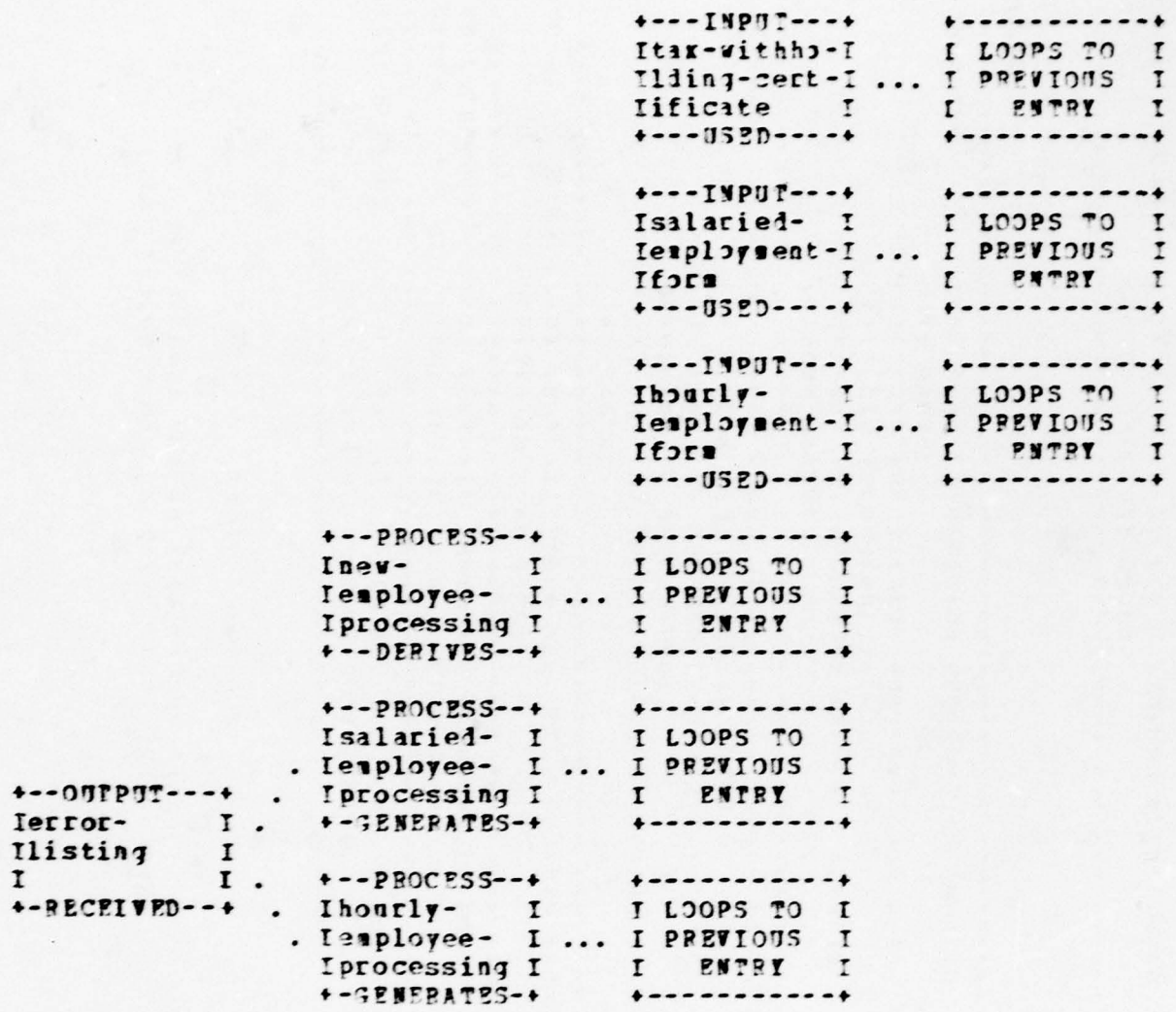
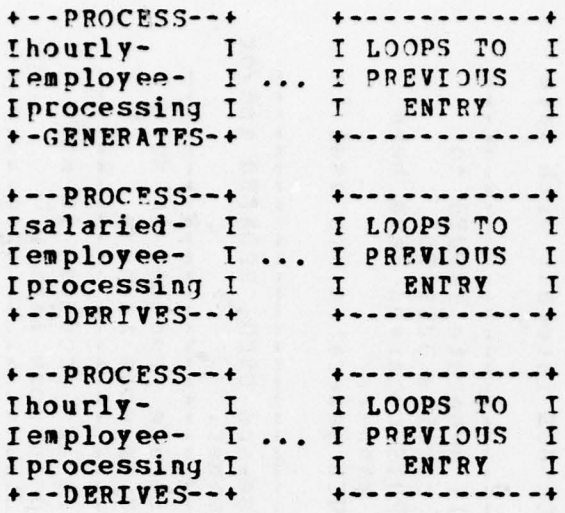


FIGURE_30

EXTENDED PICTURE



STRUCTURE:	
SET	- check that SET is broken down into ENTITIES, or INPUTS or SETS.
INPUT/OUTPUT/ GROUP/ENTITY	- check that these are eventually broken down into ELEMENTS.
GROUP/ELEMENT	- check that these are contained within some larger data structure.

FLOW:	
Process	- check that information produced is used in some manner. - check that information used has been made available (produced) in some manner. - check that all PROCESSES interact with data in some manner.
INTERFACE	- check that these all generate INPUTS to the system and/or receive OUTPUTS. - check that the OUTPUTS received have been generated in some manner. - check that the INPUTS generated are used in some manner.
SET	- check that all these are USED, UPDATED and/or DERIVED in some manner.
INPUT/OUTPUT/ ENTITY	- check that all these are produced in some manner and/or used in some manner.
GROUP/ELEMENT	- check that all these are produced in some manner and/or used in some manner.

Table 5

Completeness Checks that may be made by
Visual Analysis of the EXTENDED PICTURE Report

FORMATTED PROBLEM STATEMENT

Purpose

To present in the Language format, all description given about one or more names in an Analyzer data base.

Information Presented

The report presents, for each name used as input when generating the report, all information directly available in the data base for that name and its relationships with other names in the data base. Since this report can be generated for any type of name and the report presents all Language relationships specified for each name type, it must present all relationships as specified in Part II of the URL User's Manual.

Format

All information presented in this report is presented as legal Language statements and is formatted according to the values of the margin parameters. The margin parameters have the following effects on the format:

AMARG - indicates the column at which the first name of a name pair is to be outputted.

BMARG - indicates the column at which the second name of a name pair is to be outputted.

CMARG - specifies the number of columns between SMARG and where the text starts for a comment entry.

HMARG - indicates the column where the user defined name in a section header is to be outputted.

NMARG - indicates the column where the first name of a name list or name used in a Language statement is outputted.

RNMARG - specifies the right hand margin for names in a name list.

SMARG - indicates the column in which the Language statement headers will be started.

Figure 31 illustrates the margin parameters with respect to a part of an actual FORMATTED PROBLEM STATEMENT.

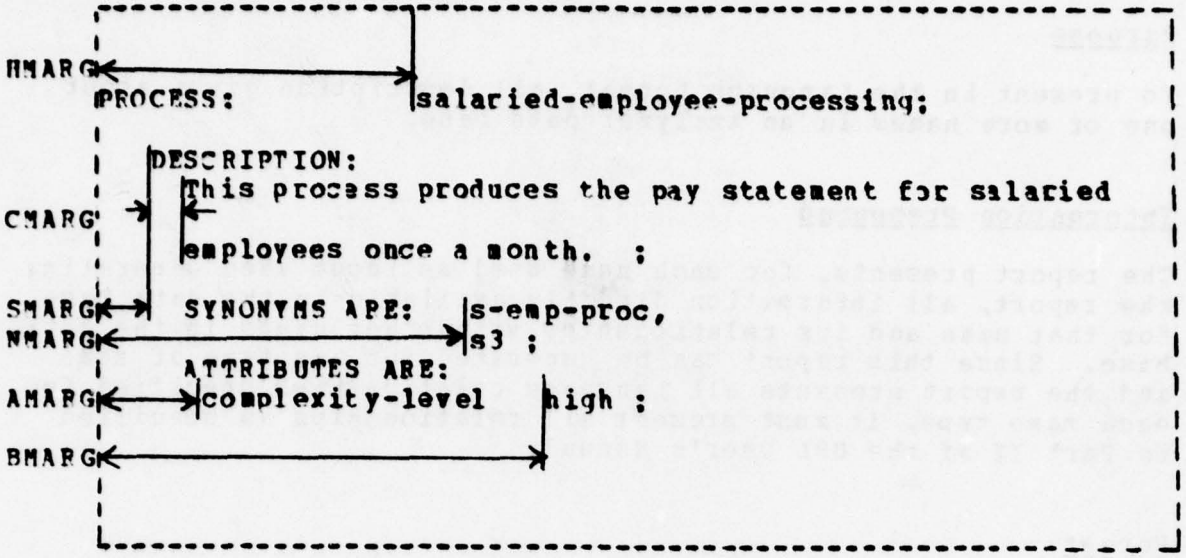


Figure 31

All Language statements presented in the FPS are numbered sequentially.

For each type of Language section, the statements within the section are ordered as given in Table 6. Sections in the FPS which describe undefined names or relationships not allowed by the syntax of the Language are presented as comment statements, i.e., preceded by the characters /* and succeeded by the characters */.

CONDITION section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	BECOMING TRUE CAUSES
	BECOMING FALSE CAUSES
	BECOMING TURE INTERRUPTS
	BECOMING FALSE INTERRUPTS
	BECOMING TRUE TERMINATES
	BECOMING FALSE TERMINATES
	BECOMING TRUE TRIGGERS
	BECOMING FALSE TRIGGERS
	BECOMING TRUE IS CALLED
	BECOMING FALSE IS CALLED
	MADE TRUE BY
	MADE FALSE BY
	TRUE WHILE
	FALSE WHILE
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6

Ordering for FORMATTED PROBLEM STATEMENT

DEFINE section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	APPLIES
	SUBSETTING-CRITERION
	MAINTAINED
	/* CONTAINED */
	/* CONNECTIVITY */
	/* CARDINALITY */
	/* HAPPENS */
	/* VALUE */
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

ELEMENT section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	CONTAINED
	SUBSETTING-CRITERION
	IDENTIFIES
	ASSOCIATED WITH
	VALUES
	USED
	UPDATED
	DERIVED
	CLASSIFICATION
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

ENTITY section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	CONSISTS
	CONTAINED
	IDENTIFIES
	RELATED
	LAYOUT
	USED
	UPDATED
	DERIVED
	CLASSIFICATION
	OCCURRENCES
	VOLATILITY
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

EVENT section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	HAPPENS
	ON INCEPTION
	ON TERMINATION
	CAUSES
	CAUSED
	INTERRUPTS
	TRIGGERS
	MAKES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

GROUP section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	CONSISTS
	CONTAINED
	SUBSETTING-CRITERION
	IDENTIFIES
	ASSOCIATED-WITH
	USED
	UPDATED
	DERIVED
	CLASSIFICATION
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

INPUT section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	GENERATED
	RECEIVED
	SUBPARTS
	PART
	CONSISTS
	CONTAINED
	LAYOUT
	USED
	CLASSIFICATION
	HAPPENS
	CAUSES
	INTERRUPTS
	TERMINATES
	TRIGGERS
	MAKES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

INTERFACE section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	GENERATES
	RECEIVES
	RESPONSIBLE
	SUBPARTS
	PART
	SECURITY-ACCESS-RIGHTS
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

INTERVAL section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	CONSISTS
	/* CONTAINED */
	/* HAPPENS */
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

MPMO section	SYNONYMS
	DESCRIPTION
	KEYWORDS
	ATTRIBUTES
	APPLIES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

OUTPUT section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	GENERATED
	RECEIVED
	SUBPARTS
	PART
	CONSISTS
	CONTAINED
	LAYOUT
	DERIVED
	CLASSIFICATION
	HAPPENS
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

PROBLEM-DEFINER section	SYNONYM
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	MAILBOX
	RESPONSIBLE
	SECURITY
	SOURCE

Table 6 (continued)

PROCESS section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	GENERATES
	RECEIVES
	SUBPARTS
	PART
	UTILIZES
	UTILIZED
	PERFORMED BY
	RESOURCE-USAGE
	USES
	UPDATES
	DERIVES
	PROCEDURE
	MAINTAINS
	SECURITY-ACCESS-RIGHTS
	HAPPENS
	INCEPTION-CAUSES
	TERMINATION-CAUSES
	INTERRUPTS
	INTERRUPTED BY
	TERMINATED
	TRIGGERS
	TRIGGERED
	MAKES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

PROCESSOR section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	GENERATES
	RECEIVES
	SUBPARTS
	PART
	CONSUMES
	PERFORMS
	UTILIZES
	UTILIZED
	PERFORMED BY
	RESOURCE-USAGE
	USES
	UPDATES
	DERIVES
	PROCEDURE
	MAINTAINS
	SECURITY-ACCESS-RIGHTS
	HAPPENS
	INCEPTION-CAUSES
	TERMINATION-CAUSES
	INTERRUPTS
	INTERRUPTED BY
	TERMINATED
	TRIGGERS
	TRIGGERED
	MAKES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

RELATION section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	ASSOCIATED-DATA
	BETWEEN
	DERIVATION
	MAINTAINED
	CONNECTIVITY
	CARDINALITY
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

RESOURCE section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	CONSUMED
	MEASURED
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

RESOURCE-USAGE-PARAMETER section	
	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	ATTRIBUTES
	RESOURCE-USAGE-PARAMETER-VALUE
	COMPUTER-PROCESSOR-CONSUMES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

```
This statement is ignored.
DESCRIPTION
SEE-MEMO
KEYWORDS
TRACE KEY
ATTRIBUTES
ASSERT
RESPONSIBLE-INTERFACE
SUBSETS
SUBSET
CONSISTS
SUBSETTING-CRITERIA
USED
DERIVED
UPDATED
DERIVATION
CLASSIFICATION
OCCURRENCES
VOLATILITY-MEMBER
VOLATILITY-SET
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE
```

Table 6 (continued)

UNITS section	SYNONYMS
	DESCRIPTION
	SEE-MEMO
	KEYWORDS
	TRACE KEY
	ATTRIBUTES
	ASSERT
	MEASURES
	RESPONSIBLE-PROBLEM-DEFINER
	SECURITY
	SOURCE

Table 6 (continued)

Options and Alternatives

Aside from the variations in the report format as the result of assignment of the margin parameters, there are a few other parameters which modify the format in some way. These parameters are given below with the effect they have on the report format.

1) NEW-LINE - specifies that the first name in a list of names for Language statements is started on the line following the statement header.

   NONEW-LINE - specifies that the names in the name list begin on the same line as the Language statement.

   For example, with NEW-LINE in effect, a Language statement would be printed as follows:

```
SYNONYMS ARE:
            s-emp-proc,
            s3;
```

   With NONEW-LINE in effect, the statements are printed as follows:

```
SYNONYMS ARE:
            s-emp-proc,
            s3;
```

2) NEW-PAGE - specifies that each section (description of single name) presented in the report would be printed beginning at the top of a new page.

   NONEW-PAGE - specifies that sections are printed one after another.

   When maintaining an up-to-date copy of the FPS, it is often desirable to generate the FPS for all names in the data base with the NEW-PAGE option. Any modifications to the description in the data base can be recorded by generating an FPS (with the NEW-PAGE option again) for those names affected by the modification.

3) ONE-PER-LINE - specifies that the names in a name list within a given statement be presented one per line.

   SEVERAL-PER-LINE - specifies that the names in a name list be presented as many as possible on a line.

Some information in the FPS can be included or left out, depending on the parameters used when generating it. Each parameter and its effect is given below.

1) COMMENT - specifies that comment statements for descriptions of undefined names and relationships not allowed by the Lanaguage syntax are to be included where applicable in the report.

NOCOMMENT - specifies that the comment statements are not printed.

2) DEFINE - specifies that descriptions for names which are described by a DEFINE section (ATTRIBUTE, ATTRIBUTE-VALUE, KEYWORD, MAILBOX, SECURITY, SOURCE, SUBSETTING-CRITERION, and SYSTEM-PARAMETER names) are included in the report when these names are given as input.

NODEFINE - specifies that the description of any name described by a DEFINE section is not presented in the report

3) DESG - specifies that the descriptions for names which are SYNONYMS for other names in the data base are presented in the report by the DESIGNATE section.

NODESG - specifies that the descriptions for names that are SYNONYMS are not presented in the report.

4) ALL-STATEMENTS(AS)
- specifies that all-statements will be printed for each section whether information was supplied or not.

NOALL-STATEMENTS(NAS)
- specifies that only those statements for which there is information contained in the data base will be printed.

5) LINE-NUMBERS(LNS)
- specifies that line numbers are to be printed on the left side of the report.

NOLINE-NUMBERS(NLNS)
- specifies that line numbers should not appear on the report.

6) PRINTEOF(PEOF) - specifies that an extra line containing

EOF is to be produced at the end of the output.

7) COMPLEMENTARY-STATEMENTS (COMP)
- specifies that complementary statements will be produced, if applicable, in each section of the report.

NOCOMPLEMENTARY-STATEMENTS((NCOMP)
- specifies that no complementary statements will be output. Thus the number of statements is the minimum necessary to describe the information in the database for this section.

For each name given as input to the command producing the report, the report presents the appropriate section to describe the name. For example, when a PROCESS name is given, it is described in a PROCESS section. Therefore, when a SYNONYM name is given as input, the report describes the SYNONYM by a DESIGNATE section rather than presenting the description of the name the SYNONYM name applies to.

An INDEX for the report is generated when the INDEX parameter is used.

The report may be generated for a single input name (via the NAME parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

For each name given as input, the software finds the name in the data base. If the name cannot be found, the message:

```
/* NAME NOT FOUND IN D.B.-                    */
```

is printed on the report. Each relationship the name has with other names is printed in the format of a legal Lanaguage statement. If no statement exists, the relationship is presented as a comment entry. Since no Language section is available to describe an UNDEFINED name, the description of the name and relationships it has with other names are presented as a comment statement.

Usage

Since the FPS presents all the description given about each name in the data base, the report is beneficial in checking the accuracy of each description. It is usually recommended that an FPS for all names be maintained as a reference and updated when changes are made to the data base.

Examples

Figure 32 presents an FPS for a single name. This example was generated by the following command:

```
FORMATTED-PROBLEM-STATEMENT NAME=payroll-processing
```

Figure 33 presents the report for all ENTITY names defined in a particular data base. This example was generated by the following commands:

```
NAME-GEN S='ENTITY'
FPS
```

FIGURE_32

FORMATTED PROBLEM STATEMENT

```
PARAMETERS FOR: PPS

   NAME=payroll-processing NOINDEX PRINT EMPTY NOPUNCH SMARG=5 MMARG=20 AMARG=10 BMARG=25
   RMMARG=70 CMARG=1 HMARG=40 NODESIGNATE ONE-PER-LINE DEFINE COMMENT NONEW-PAGE NONEW-LINE
   NOALL-STATEMENTS COMPLEMENTARY-STATEMENTS LINE-NUMBERS PRINTED

 1 PROCESS                                   payroll-processing;
 2     SYNONYMS ARE:  payproc,
 3                    p1;
 4     DESCRIPTION;
 5           This process represents the highest level process
 6           in the target system.  it accepts and processes
 7           all inputs and produces all outputs.;
 8     SEE-MEMO:      goals-memo,
 9                    process-memo;
10     GENERATES:     paysystem-outputs;
11     RECEIVES:      employee-information;
12     SUBPARTS ARE:  new-employee-processing,
13                    terminating-emp-processing,
14                    hourly-employee-processing,
15                    salaried-employee-processing,
16                    process-library;
17     PERFORMED BY:  payroll-processor;
18     USES:          employee-information,
19                    payroll-master-information;
20     UPDATES:       payroll-master-information;
21     DERIVES:       paysystem-outputs;
22     HAPPENS:
23          no-of-payroll-processing
24           TIMES-PER      month;
25     RESPONSIBLE-PROBLEM-DEFINER IS:
26                    michel-j-bastarache;
27
28 EOF EOF EOF EOF EOF
```

FIGURE_33

FORMATTED PROBLEM STATEMENT

```
PARAMETERS FOR: FPS

   FILE NOINDEX PRINT EMPTY NOPUNCH SMARG=5 NMARG=20 AMARG=10 BMARG=25 BNMARG=70 CMARG=1 HMARG=40
   NODESIGNATE ONE-PER-LINE DEFINE COMMENT NONEW-PAGE NONEW-LINE NOALL-STATEMENTS
   COMPLEMENTARY-STATEMENTS LINE-NUMBERS PRINTTOP

    1 ENTITY                                    department-information;
    2     SYNONYMS ARE:  dept-info;
    3     DESCRIPTION;
    4           This information holds all current data relevent about each
    5           department;
    6     CONSISTS OF:
    7                        department,
    8          no-of-supervisors
    9                        supervisor,
   10                        number-of-employees,
   11                        total-budget,
   12                        remaining-funds;
   13     CONTAINED IN:  payroll-master-information,
   14                    department-file;
   15     IDENTIFIED BY: department;
   16     /* LEFT */ RELATED TO:
   17          hourly-employee-information
   18           VIA           dept-hourly-emp-relation;
   19     /* LEFT */ RELATED TO:
   20          salaried-employee-information
   21           VIA           dept-salaried-emp-relation;
   22     OCCURRENCES:   no-of-departments;
   23
   24 ENTITY                                   hourly-employee-information;
   25     SYNONYMS ARE:  h-emp-info;
   26     DESCRIPTION;
   27           This information holds all current data about each hourly
   28           employee.;
   29     CONSISTS OF:
   30                        employee-name,
   31                        employee-identification-number,
   32                        social-security-number,
   33                        pay-grade-code,
   34                        address,
   35                        phone,
```

FIGURE_33

FORMATTED PROBLEM STATEMENT

```
36                      employment-date,
37                      number-of-deductions,
38                      department,
39                      cumulative-gross-pay,
40                      cumulative-federal-deductions,
41                      cumulative-state-deductions,
42                      cumulative-fica-deductions,
43                      age,
44                      sex,
45                      status-code,
46                      employment-status,
47                      cumulative-hours;
48      CONTAINED IN:  payroll-master-information,
49                     hourly-employee-file;
50      IDENTIFIED BY: employee-identification-number;
51      /* RIGHT */ RELATED TO:
52           department-information
53            VIA           dept-hourly-emp-relation;
54      USED BY:       hourly-employee-processing;
55      USED BY:
56           term-report-entry-generation
57            TO DERIVE     term-report-entry;
58      DERIVED BY:    hourly-information-creation;
59      CLASSIFICATION IS:
60           classified     2,
61           secret         1,
62           top-secret     2,
63           limited-security
64                          3;
65      OCCURRENCES:   no-of-hourly-employees;
66      VOLATILITY;
67           This information is changed about once a week .;
68      SECURITY IS:   no-restrictions;
69
70  ENTITY                                  salaried-employee-information;
71      SYNONYMS ARE:  s-emp-info;
72      DESCRIPTION;
73           This information holds all current data about salaried
74           employees.;
75      CONSISTS OF:
76                      employee-name,
```

152c

FIGURE_33

FORMATTED PROBLEM STATEMENT

```
                        employee-identification-number,
                        social-security-number,
                        pay-grade-code,
                        address,
                        phone,
                        employment-date,
                        number-of-deductions,
                        department,
                        cumulative-federal-deductions,
                        cumulative-gross-pay,
                        cumulative-state-deductions,
                        cumulative-fica-deductions,
                        age,
                        sex,
                        employment-status,
                        status-code;
    CONTAINED IN:  payroll-master-information,
                   salaried-employee-file;
    IDENTIFIED BY: employee-identification-number;
    /* RIGHT */ RELATED TO:
         department-information
          VIA           dept-salaried-emp-relation;
    USED BY:       salaried-employee-processing;
    USED BY:
         term-report-entry-generation
          TO DERIVE     term-report-entry;
    DERIVED BY:    salaried-information-creation;
    OCCURRENCES:   no-of-salaried-employees;
    VOLATILITY;
         This information is changed about once a month.;
    SECURITY IS:   no-restrictions;

EOF EOF EOF EOF EOF
```

FREQUENCY REPORT

Purpose

To present all information based on the use of the HAPPENS statement in a particular Analyzer data base.

Information Presented

The report presents size and volume information about all INPUT, OUTPUT, PROCESS and EVENT names defined in the data base with respect to the HAPPENS statements connected to those types of names. An entry is made in the report for each INPUT, OUTPUT, PROCESS and EVENT with a HAPPENS statement and the entries are grouped by the INTERVAL over which the HAPPENS statement is effective. SYSTEM-PARAMETERS are contained in each entry to define the number of times the name associated to the entry occurs within the INTERVAL.

Format

The report presents all HAPPENS statements relative to a specific INTERVAL whose name is printed on a heading. For each INTERVAL, three headings are printed and the frequency information pertaining to the particular INTERVAL is listed below these headings. The headings are:

NAME	-	the names of the INPUTS, OUTPUTS, PROCESSES and EVENTS which HAPPEN within the designated INTERVAL are listed.
TYPE	-	the name type of each of the names given under NAME is listed.
TIMES HAPPENS	-	the SYSTEM-PARAMETER or number used in the HAPPENS statement for each of the names given under NAME is listed.

The INTERVALS are presented alphabetically in the report. No ordering is imposed on the names listed.

Options and Alternatives

No options are available for this report.

Analysis

The data base is searched for an INTERVAL name that is connected to a HAPPENS relationship. If none are found, the message:

URA286:PRINTV: NO FREQUENCY INFORMATION IN DATA BASE

is printed. If an INTERVAL name is found, it is printed out and for each HAPPENS statement connected to it the name HAPPENS statement applies to, its name type and associated SYSTEM-PARAMETER are retrieved and printed out.

The search for another INTERVAL is continued until all names have been tested.

Usage

The report is helpful to analysts in checking that all items in the description which are to be logically related via their frequency are grouped together.

The report is also beneficial to system designers when considering the relationships of various parts of system with respect to frequency, and the amount of input and output to be handled by the target system.

Examples

Figure 34 presents the FREQUENCY REPORT for a Language description of a target system.

FIGURE_34

FREQUENCY REPORT

INTERVAL: month

NAME	TYPE	TIMES HAPPENS
employment-termination-form	INPUT	several
error-listing	OUTPUT	no-of-payroll-processing
hourly-employment-form	INPUT	several
salaried-employee-report	OUTPUT	no-salaried-emp-processing
salaried-employment-form	INPUT	several
tax-withholding-certificate	INPUT	several
payroll-processing	PROCESS	no-of-payroll-processing
salaried-employee-processing	PROCESS	no-salaried-emp-processing
salaried-emp-processing-init	EVENT	no-salaried-emp-processing

INTERVAL: week

NAME	TYPE	TIMES HAPPENS
hired-employee-report	OUTPUT	no-new-emp-processing
hourly-employee-report	OUTPUT	no-hourly-emp-processing
time-card	INPUT	no-of-hourly-employees
terminated-employee-report	OUTPUT	no-terminating-emp-processing
new-employee-processing	PROCESS	no-new-emp-processing
hourly-emp-processing	PROCESS	no-hourly-emp-processing
terminating-emp-processing	PROCESS	no-terminating-emp-processing
hourly-emp-processing-init	EVENT	no-hourly-emp-processing
new-employee-processing-init	EVENT	no-new-emp-processing
termination-processing-init	EVENT	no-terminating-emp-processing
hourly-employee-processing	PROCESS	1
hourly-paycheck-validation	PROCESS	50
hourly-emp-update	PROCESS	no-of-hourly-employees
h-report-entry-generation	PROCESS	no-of-hourly-employees
hourly-paycheck-production	PROCESS	no-of-hourly-employees
time-card-validation	PROCESS	50
pay-computation-validation	PROCESS	50
hours-update	PROCESS	no-of-hourly-employees
gross-pay-update	PROCESS	no-of-hourly-employees
federal-deductions-update	PROCESS	no-of-hourly-employees
state-deductions-update	PROCESS	no-of-hourly-employees
fica-deductions-update	PROCESS	no-of-hourly-employees
funds-update	PROCESS	no-of-hourly-employees
h-gross-pay-computation	PROCESS	no-of-hourly-employees

FIGURE_34

FREQUENCY REPORT

NAME	TYPE	TIMES HAPPENS
total-hours-computation	PROCESS	no-of-hourly-employees
total-deductions-computation	PROCESS	no-of-hourly-employees
tax-computation	PROCESS	no-of-hourly-employees
net-pay-computation	PROCESS	no-of-hourly-employees

IDENTIFIER INFORMATION REPORT

Purpose

To present all information based on the use of IDENTIFIERS for ENTITIES in a particular Analyzer data base.

Information Presented

If IDENTIFIER names are used as input when generating this report (and the IDENTIFIER parameter is specified), those ENTITIES which the input names IDENTIFY are presented in the report.

If ENTITY names are used as input when generating the report (and the ENTITY parameter is specified), those IDENTIFIERS which the input names are IDENTIFIED by are presented in the report.

In either case, the information is presented as a matrix. An analysis of the information in the matrix produces some statistics showing the number of IDENTIFIERS each ENTITY in the matrix had and the number of ENTITIES each IDENTIFIER identifies in the matrix.

Format

If the IDENTIFIER parameter is used when generating the report, any names given as input which do not IDENTIFY any ENTITY in the data base are flagged at the beginning of the report. If the ENTITY parameter is used when generating the report, any names given as input which are not IDENTIFIED by any IDENTIFIER in the data base are flagged at the beginning of the report.

Two lists of names are then presented, one labeled ROW NAMES and the other COLUMN NAMES. If the IDENTIFIER parameter was used when generating the report, the names designated as ROW NAMES are those which were given as input. If the ENTITY parameter was used when generating the report, the names designated as COLUMN NAMES are those which were given as input.

In any case, each name under ROW NAMES IDENTIFIES one or more names under COLUMN NAMES and each name under COLUMN NAMES is IDENTIFIED by one or more names under ROW NAMES.

A matrix is then printed to show the relationships between the names designated as ROW NAMES (which are represented by the rows of the matrix) and the names designated as COLUMN NAMES (which are represented by the columns of the matrix). The rows and columns of the matrix are numbered to correspond to the number assigned to each name in the list of ROW NAMES and COLUMN NAMES, respectively.

An asterisk (*) entry at the intersection of a particular row and column of the matrix designates that the name represented by the row IDENTIFIES the ENTITY represented by the column.

Inspection of an entire row reveals all ENTITIES that a particular name (represented by the row) IDENTIFIES. Inspection of an entire column reveals all IDENTIFIERS for the particular name represented by the column.

A summary section is also included in the report presenting for each ROW NAME:

- The row it was represented by in the matrix (ROW).
- Its name type (TYPE).
- The number of * entries in its row (or the number of ENTITIES it IDENTIFIES) (COUNT).

The summary presents for each COLUMN NAME:

- The column it was represented by (COLUMN).
- Its name type (TYPE).
- The number of * entries in its column (or the number of IDENTIFIERS for it) (COUNT).

The summary section for ROW and COLUMN names is ordered in decreasing order of COUNT.

Options and Alternatives

The report must be generated using either the IDENTIFIER or ENTITY parameter. If the IDENTIFIER parameter is used, all names given as input must be IDENTIFIER names. If the ENTITY parameter is used, all names given as input must be ENTITY names.

The report may be generated for a single input name (via the NAME parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

For each name given as input, the software finds the name in the data base. If the name is not found, the message:

```
URA099:IDENTR: NAME NOT IN D.B. -                    or
URA052:IDENTC: NAME NOT IN D.B. -
```

is printed depending on whether the IDENTIFIER or ENTITY parameter was specified for the command, respectively.

If the IDENTIFIER parameter is used, each name given as input is checked that it IDENTIFIES one or more ENTITIES. If it does not, the name is listed under the message:

URA304:IDENTR: THE FOLLOWING NAMES DO NOT IDENTIFY ANYTHING

The ENTITIES that are IDENTIFIED by the input names are found. The list of all input names and the list of all ENTITIES retrieved is then printed.

If the ENTITY parameter is used, each name given as input is checked that it is IDENTIFIED by one or more IDENTIFIERS. If it does not, the name is listed under the message:

URA308:IDENTC: THE FOLLOWING NAMES ARE NOT IDENTIFIED BY ANYTHING

The IDENTIFIERS for the ENTITIES given as input are found. The list of all input names and the list of all IDENTIFIERS retrieved is then printed.

A matrix is printed out to illustrate the relationships between the names in the two lists and each relationship is designated by an asterisk.

A summary is then produced by counting the number of asterisks appearing in each row and column of the matrix.

Usage

The report presents information that aids the analyst in checking the completeness and consistency of the problem statement by:

1) - identifying those ENTITIES which do not have IDENTIFIERS. This can be accomplished by the following Analyzer commands:

```
NAME-GEN S='ENTITY'
ENTITY-IDENTIFIER ENTITY
```

2) - being in an easy-to-analyze format to check that the IDENTIFIERS defined for the problem statement are being used properly. For example, a typing error may result in an IDENTIFIER being used in the wrong context.

The report aids the system designer by presenting those ENTITIES with the same IDENTIFIERS and aids in determining a consistent and well-defined identifier coding structure.

Example

Figure 35 presents the report using the ENTITY parameter. The Analyzer commands used to generate this example were:

```
NAME-GEN S='ENTITY'
ENTITY-IDENTIFIER ENTITY
```

FIGURE_35

IDENTIFIER INFORMATION REPORT

PARAMETERS FOR: EI

FILE ENTITY

ROW NAMES		COLUMN NAMES	
1 department	ELEMENT	1 department-information	ENTITY
2 employee-identification-number	ELEMENT	2 hourly-employee-information	ENTITY
		3 salaried-employee-information	ENTITY

THE ROWS ARE IDENTIFIERS OF THE COLUMNS WITH *S

```
   123
  +---+
1 I*  I
2 I **I
  +---+
```

FIGURE_35

IDENTIFIER INFORMATION REPORT

THE NUMBER OF COLUMNS IDENTIFIED BY THE ROWS

	ROW	TYPE	COUNT
2	employee-identification-number	ELEMENT	2
1	department	ELEMENT	1

THE NUMBER OF ROWS THAT IDENTIFY THE COLUMNS

	COLUMN	TYPE	COUNT
1	department-information	ENTITY	1
2	hourly-employee-information	ENTITY	1
3	salaried-employee-information	ENTITY	1

KWIC INDEX

Purpose

To present, in an easy to inspect format, logical groupings of names defined in a particular Analyzer data base with respect to the spelling of the names.

Information Presented

The report presents, for all those names used as input, an alphabetical listing consisting of an entry for each name as it appears as input and entries for each permutation of the name (about the dashes). For example, if the name hourly-employee-form was supplied as input, there would be entries for:

```
employment-form                          hourly
form                                     hourly-employment
hourly-employment-form
```

in the report. When there are several names used as input then all names with the word "employment" in them would have entries group together, all those with "form" would be grouped together, etc.

Format

The entries in the report are ordered alphabetically and numbered sequentially. There are two parts of each entry, the right hand side of the entry presents that part of the user defined name that has been stripped off for a permutation of the name and the left hand side of the entry presents the remaining part of the name. The distance (the number of columns) between the right and left sides of the entry can be varied by the value assigned to the DIF parameter.

Options and Alternatives

The DIF parameter may take on any value from 2 to 52.

Analysis

Each name given as input to the software generating the report is inspected, separated at the dashes in the name and a list is formed consisting of the original name and all permutations of the name.

After all names in the input list have been processed, the newly

formed list is sorted and presented as the report.

Usage

The KWIC INDEX aids analysts in maintaining name conventions used in the target system description process and for finding names in the description based on the keywords within the names.

It is often desirable to use some conventions in assigning names to objects defined in a target system description and the KWIC INDEX aids in maintaining these. For example, by issuing the following Analyzer commands:

```
NAME-GEN S='ELEMENT'
KWIC
```

a KWIC INDEX is presented for all ELEMENT names so that consistency of naming can be checked.

Examples

Figure 36 presents a KWIC INDEX for INPUT names defined in a problem statement.

FIGURE_36

KWIC INDEX

PARAMETERS FOR: KWIC

DIF=20 FILE

SEQ	N A M E (PERMUTED)	
1	card	time
2	certificate	tax-withholding
3	employee-information	
4	employment-form	hourly
5	employment-form	salaried
6	employment-termination-form	
7	form	hourly-employment
8	form	salaried-employment
9	form	employment-termination
10	hourly-employment-form	
11	information	employee
12	salaried-employment-form	
13	tax-withholding-certificate	
14	termination-form	employment
15	time-card	
16	withholding-certificate	tax

NAME-GEN

Purpose

To select all names from the data base with respect to some selection criteria (as specified by parameters).

Information Presented

The report presents a list of names and their corresponding name types. The types of names in the list are specified by the selection criteria used with the SELECTION parameter. The selection criteria is specified by using logical AND, OR, NOT operators with various operands (including all name types) in a boolean expression. For example, when the SELECTION is INTERFACE OR PROCESS, the report presents all INTERFACE and PROCESS names in the data base. When the SELECTION is PROCESS AND KEYWORD=level-2, the report presents all PROCESSES in the data base that have the KEYWORD "level-2" associated as part of their description.

Format

Any entry in the report is printed for each name retrieved and consists of:

- the name retrieved, and
- the name type of the name

The entries in the report are ordered in one of three ways:

1) alphabetically on the names within a name type (which are also ordered alphabetically) when the ORDER=BYTYPE parameter is in effect.

2) alphabetically on the names when the ORDER=ALPHA parameter is in effect.

3) on the attribute values associated with specified attribute names when user attribute names are specified. The output will be sorted on the value of the first attribute name, then on the second name, etc. Names which do not have the specified ATTRIBUTE are placed at the beginning of the list of names.

All ORDER options except ALPHA may be used in any combination in the list of parameters. The list of parameters may not exceed five options. The BYTYPE option may appear only once. The names are ordered first on the first option in the ordering list; within that ordering on the second option in the ordering list, etc.

Options and Alternatives

The Selection Criteria parameter permits the user to specify the types of names to be retrieved by giving a boolean expression. Either the SELECTION or INPUT parameter must be used to give the boolean expression. If SELECTION='boolean expression' is specified, the boolean expression appears inside the quotes following the equal sign. If INPUT=fdname is used, the boolean expression must be in the file specified by the user.

The names retrieved in the NAME GFN report depend on the items or operands given in the boolean expression. Each of these operands represents some grouping of names which may be contained in the data base. An explanation of each of these operands is given below.

The following name types may be used as operands:

ATTRIBUTE	INTERFACE	RESOURCE
ATTRIBUTE-VALUE	INTERVAL	RESOURCE-USAGE-PARAMETER
CLASSIFICATION	KEYWORD	SECURITY
CONDITION	MAILBOX	SET
ELEMENT	MEMO	SOURCE
ENTITY	OUTPUT	SUBSETTING-CRITERION
EVENT	PROBLEM-DEFINER	SYSTEM-PARAMETER
GROUP	PROCESS	UNDEFINED
INPUT	PROCESSOR	UNIT

ALL

When the ALL operand is specified, the names of all name types except SYNONYM and UNDEFINED will be presented.

TOTAL

When the TOTAL operand is specified, every name in the data base will be presented.

BASIC

When the BASIC operand is specified, the basic names will be included in the output. The "Basic" names are those names which are not SYNONYMS.

UNDEFINED

When the UNDEFINED operand is specified, the undefined names will be presented.

ATTR=attr-name[,value]

When the ATTR operand is specifed, those names with the given user-name as an ATTRIBUTE are selected to be part of the output. The user-name must be a name defined as an ATTRIBUTE in the data base.

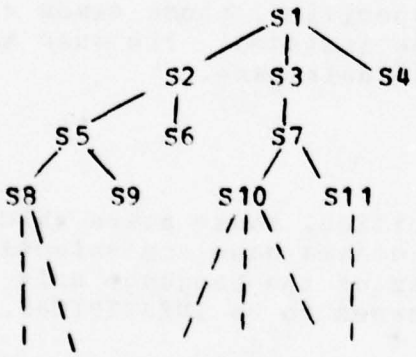
If an ATTRIBUTE-VALUE is specified as part of the user name then only those names with the given user-name as an ATTRIBUTE-VALUE, for the ATTRIBUTE designated by the ATTR parameter, are selected to be part of the output. The user-name must be an ATTRIBUTE-VALUE name in the data base.

SUBPARTS-OF(SO)=user-name[,level]

All names which belong to the SUBPARTS structure for a given name (as would be retrieved for the STRUCTURE report) can be retrieved by specifying:

SUBPARTS-OF=name

where the name is an INPUT, OUTPUT, PROCESS or INTERFACE name which has SUBPARTS information defined for it. The number of levels to go down and retrieve names to present in the report is specified by the SUBLEVEL parameter or by attaching a comma and a level number after the user-name with the SUBPARTS-OF parameter. If SUBLEVEL=0, then all levels of names are presented. If SUBLEVEL=1, then only those names which are PART OF the SUBPARTS OF name are presented. The following picture may clarify the association between the value of SUBLEVEL and the names presented.



SUBPARTS-OF name

SUBLEVEL=1

SUBLEVEL=2

SUBLEVEL=3

SUBLEVEL=0

Generation of the report with SUBPARTS-OF=S1 and SUBLEVEL=3 would present S2, S3, S4, S5, S6, S7, S8, S9, S10 and S11 in the report. Generation of the report with SUBPARTS-OF=S1 and SUBLEVEL=1 would present the names S2, S3 and S4.

SYNONYMS

When the SYNONYMS operand is specified, all SYNONYMS are presented for each name retrieved in the report in addition to the basic form of the name. If only the SYNONYMS are desired, the basic names may be suppressed by specifying the NOBASIC and SYNONYM parameters. With standard defaults in effect, the BASIC and NOSYNONYM parameters are used.

KEY=user-name

When the KEY operand is specified, those names with the given user-name as a KEYWORD are selected to be part of the output. The user-name must be a name defined as a KEYWORD in the data base.

PD=user-name

When the PD operand is specified, those names with the given user-name as a PROBLEM-DEFINER are selected to be part of the output. The user-name must be a name defined as a PROBLEM-DEFINER in the data base.

SOURCE=user-name

When the SOURCE operand is specified, those names with given user-name as a SOURCE will be included in the output. The user-name must be defined as a SOURCE in the data base.

SECURITY=user-name

When the SECURITY operand is specified, those names with given user-name as a SECURITY will be included. The user name must be defined as SECURITY name in the data base.

USAGE=user-name

When the usage operand is specified, those names which have user-name as IDENTIFIERS in the data base are selected to be part of the output. The syntax of the Language only allows ELEMENT, GROUP and UNDEFINED names to be IDENTIFIERS.

The names retieved in the NAME GEN report also depend on the operators which are combined with the operands to form the boolean expression. These operators further define grouping of names when combined with operands. An explanation of each of these operators is given below.

NOT, ¬ ,/

The NOT operator placed before an operand specified that the names associated with that operand will not be retrieved. For example, a boolean expression of the form NOT PROCESS means that all names in the data base except for PROCESS names will be retrieved.

AND,&,*

The AND operator specifies that any name retrieved from the data base must meet the criterion designated before and after the AND operator. For example, a boolean expression of the form PROCESS AND KEY=level-1 means that any name retrieved must be a PROCESS name as well as have the KEYWORD "level-1" attached to it.

OR,|,+

The OR operator specifies that any name retrieved from the data base must either meet the criterion designated before the OR operator, or after the operator, or both. For example, a boolean expression of the form PROCESS OR KEY=level-1 means that any name retrieved must be either a PROCESS name, or have the KEYWORD "level-1", or be both a PROCESS and have the KEYWORD "level-1".

Analysis

Each name defined in the data base is checked against the parameters for the command. If it satisfies the requirements as specified by the parameters, it is placed in a list. After all names in the data base have been checked, the list is sorted as the report.

If NAME-GEN is generated for an empty data base, the message:

URA049: ENUMRT: NO NAMES IN DATA BASE

will be printed.

If there are no names in the data base which satisfy the selection criteria, the message:

URA523: GETNML: NO NAMES WHICH MATCH CRITERION

will be printed.

If the selection string given as input is not a legal boolean expression, the message:

URA526: NGPRS: INVALID SELECTION STRING

will be printed.

If the selection string given as input contains an illegal operand or operator, the message:

URA527: NGPRS: INVALID ITEM IN SELECTION STRING

will be printed.

If more than five ORDER parameters have been given or the ORDER parameters have been given incorrectly, the message:

URA528: PREPAR: TOO MANY ORDER PARAMETERS

will be printed.

If the name given in the order list is not an ATTRIBUTE, the message:

URA529: PREPAR: NAME IN ORDER LIST NO ATTRIBUTE

will be printed.

If too many levels have been specified via the SUBPARTS-OF or SUBLEVEL parameters (a fifty levels is maximum), the message:

URA535: NGPRS: TOO MANY LEVELS, MAX OF 50 ALLOWED

will be printed.

Usage

It is an important aid to the analyst in obtaining other reports and outputs. For example, the analyst can ask for a list of all SET, ENTITY and GROUP names and with this list then ask for a CONTENTS REPORT for these names.

It is also used by the analyst as a reference to what names have been used and how they have been used (i.e., what their name types are).

The output can also be used effectively by project management to measure productivity of the project members. This can be done by retrieving a list of all names in the data base defined by a particular problem definer (analyst) and comparing it to previous lists.

Finally, the NAME GEN output can become an integral part of the final specifications as it acts as a directory in specifying name lists corresponding to certain selection criteria (a directory of all data elements may be desired before a section which deals with the definition of each element in detail).

Examples

Figure 37 presents a NAME GEN report produced for all PROCESS names which have "level-2" defined as one of their KEYWORDS. The command used to generate this example was:

```
NAME-GEN S='KEY=terminal AND PROCESS'
```

Figure 38 presents a NAME GEN report produced for all names which are not PROCESSES and which have "level-2" defined as one of their KEYWORDS. The command used to generate this example was:

```
NAME-GEN S='PROCESS * NOT KEY=terminal'
```

Figure 39 presents a NAME GEN report produced for all names which are SUBPARTS OF security-control-input and which have the ATTRIBUTE occurrences with the ATTRIBUTE-VALUE unscheduled. The command used to generate this example was:

```
NAME-GEN S='ATTR=occurrence-type, unscheduled
```

Figure 40 presents a NAME GEN report produced for all INPUT and OUTPUT names which have the ATTRIBUTE copies or arrival-type with the ATTRIBUTE-VALUEs 3 or random. The command used to generate this example was:

```
NAME-GEN S='{INPUT|OUTPUT}&{ATTR=copies,3|arrival-type,random}'
```

FIGURE_37

Name Generation

```
PARAMETERS FOR: NG

   PRINT PUNCH EMPTY SELECTION='KEY=terminal AND PROCESS' ORDER=BYTYPE

    1    federal-deductions-update          PROCESS
    2    fica-deductions-update             PROCESS
    3    funds-update                       PROCESS
    4    gross-pay-update                   PROCESS
    5    h-gross-pay-computation            PROCESS
    6    hours-update                       PROCESS
    7    net-pay-computation                PROCESS
    8    pay-computation-validation         PROCESS
    9    s-gross-pay-computation            PROCESS
   10    state-deductions-update            PROCESS
   11    tax-computation                    PROCESS
   12    time-card-validation               PROCESS
   13    total-deductions-computation       PROCESS
   14    total-hours-computation            PROCESS
```

FIGURE_38

Name Generation

PARAMETERS FOR: NG

```
   PRINT PUNCH EMPTY SELECTION='PROC * NOT KEY=terminal' ORDER=BY TYPE

    1    department-file-addition            PROCESS
    2    department-file-removal             PROCESS
    3    h-report-entry-generation           PROCESS
    4    hire-report-entry-generation        PROCESS
    5    hourly-emp-processing               PROCESS
    6    hourly-emp-update                   PROCESS
    7    hourly-employee-processing          PROCESS
    8    hourly-information-creation         PROCESS
    9    hourly-information-deletion         PROCESS
   10    hourly-paycheck-production          PROCESS
   11    hourly-paycheck-validation          PROCESS
   12    new-employee-processing             PROCESS
   13    payroll-processing                  PROCESS
   14    process-library                     PROCESS
   15    s-report-entry-generation           PROCESS
   16    salaried-emp-update                 PROCESS
   17    salaried-employee-processing        PROCESS
   18    salaried-information-creation       PROCESS
   19    salaried-information-deletion       PROCESS
   20    salaried-paycheck-production        PROCESS
   21    salaried-paycheck-validation        PROCESS
   22    term-report-entry-generation        PROCESS
   23    terminating-emp-processing          PROCESS
```

FIGURE_39

Name Generation

```
PARAMETERS FOR: NG

   PRINT PUNCH EMPTY SELECTION='ATTR=occurrence-type,unscheduled' ORDER=BYTYPE

   1    validation                          RESOURCE-USAGE-PARAMETER
```

FIGURE_40

Name Generation

PARAMETERS FOR: NG

```
PRINT PUNCH EMPTY SELECTION='{INP|OUTPUT}&{ATTR=arrival-type,random | ATTR=copies,3}'
ORDER=BYTYPE
```

1	employment-termination-form	INPUT
2	hourly-employment-form	INPUT
3	salaried-employment-form	INPUT
4	tax-withholding-certificate	INPUT
5	hourly-employee-report	OUTPUT
6	salaried-employee-report	OUTPUT

NAME LIST

Purpose

To present a list of all names defined in a particular Analyzer data base, the name type associated with the name and SYNONYMS defined for each name.

Information Presented

The report presents every name currently defined in the user's data base, the name type associated with each name and the SYNONYMS associated with each name

Format

An entry in the report is printed for each name in the Analyzer data base and consists of:

- the name,
- the name type of the name, and
- any SYNONYMS for the name,

which are listed under the headings: NAME, TYPE, and SYNONYM, respectively.

The entries within the report are ordered in one of two ways: alphabetically on the names when the ORDER=ALPHA parameter is in effect and, alphabetically on the names within name type (which are also ordered alphabetically) when the ORDER=BYTYPE parameter is in effect.

If no SYNONYMS are available for a particular name a dash (-) is printed under the SYNONYM heading. If more than one SYNONYM exists for a name, they are listed beneath each other.

Options and Alternatives

No options other than the ORDER=ALPHA and ORDER=BYTYPE features are available for this report.

Analysis

Each name in the data is inspected and its name type and any SYNONYMS for the name are retrieved. After this information has been collected for all names in the data, it is sorted and presented as the report.

Usage

The report is intended to be used as a directory facility by anyone needing a reference including all names defined in the data base.

Examples

Figure 41 presents the NAME LIST report generated with the ORDER=BYTYPE option.

PIGDRB_41

NAME LIST

PARAMETERS FOR: NL

ORDER=BYTYPE

	NAME	TYPE	SYNONYM
1	error-flags	*** UNDEFINED ***	-
2	error-messages	*** UNDEFINED ***	-
3	arrival-type	ATTRIBUTE	-
4	color	ATTRIBUTE	-
5	complexity-level	ATTRIBUTE	-
6	copies	ATTRIBUTE	-
7	data-standard	ATTRIBUTE	-
8	number-of-lines	ATTRIBUTE	-
9	occurrence-type	ATTRIBUTE	-
10	processor-type	ATTRIBUTE	-
11	type	ATTRIBUTE	-
12	character	ATTRIBUTE-VALUE	-
13	date	ATTRIBUTE-VALUE	-
14	high	ATTRIBUTE-VALUE	-
15	human	ATTRIBUTE-VALUE	-
16	low	ATTRIBUTE-VALUE	-
17	medium	ATTRIBUTE-VALUE	-
18	numeric	ATTRIBUTE-VALUE	-
19	random	ATTRIBUTE-VALUE	-
20	scheduled	ATTRIBUTE-VALUE	-
21	unscheduled	ATTRIBUTE-VALUE	-
22	white	ATTRIBUTE-VALUE	-
23	classified	CLASSIFICATION	-
24	limited-security	CLASSIFICATION	-
25	secret	CLASSIFICATION	-
26	top-secret	CLASSIFICATION	-
27	all-data-for-employee-found	CONDITION	-
28	employee-id-valid	CONDITION	-
29	hours-update-finished	CONDITION	-
30	no-time-card-for-employee	CONDITION	-
31	time-card-for-employee	CONDITION	-
32	time-cards-ready	CONDITION	-
33	age	ELEMENT	-
34	apartment-number	ELEMENT	-

FIGURE_41

NAME LIST

	NAME	TYPE	SYNONYM
35	birthdate	ELEMENT	-
36	check-number	ELEMENT	-
37	city	ELEMENT	-
38	count-of-hourly-employees	ELEMENT	-
39	count-of-salaried-employees	ELEMENT	-
40	cumulative-federal-deductions	ELEMENT	-
41	cumulative-fica-deductions	ELEMENT	-
42	cumulative-gross-pay	ELEMENT	-
43	cumulative-hours	ELEMENT	-
44	cumulative-state-deductions	ELEMENT	-
45	cumulative-tax-deductions	ELEMENT	-
46	current-date	ELEMENT	-
47	department	ELEMENT	-
48	employee-identification-number	ELEMENT	-
49	employment-date	ELEMENT	-
50	employment-status	ELEMENT	-
51	error-code	ELEMENT	-
52	federal-tax	ELEMENT	-
53	fica-tax	ELEMENT	-
54	first-name	ELEMENT	-
55	gross-pay	ELEMENT	-
56	hours-per-day	ELEMENT	-
57	house-number	ELEMENT	-
58	initial	ELEMENT	-
59	job-number	ELEMENT	-
60	job-title	ELEMENT	-
61	net-pay	ELEMENT	-
62	number-of-deductions	ELEMENT	-
63	number-of-employees	ELEMENT	-
64	overtime-hours-worked	ELEMENT	-
65	pay-date	ELEMENT	-
66	pay-grade-code	ELEMENT	-
67	pay-rate	ELEMENT	-
68	phone	ELEMENT	-
69	regular-hours-worked	ELEMENT	-
70	remaining-funds	ELEMENT	-
71	salary	ELEMENT	-
72	sex	ELEMENT	-
73	social-security-number	ELEMENT	-

FIGURE_41

NAME LIST

	NAME	TYPE	SYNONYM	
74	state	ELEMENT		-
75	state-tax	ELEMENT		-
76	status-code	ELEMENT		-
77	street	ELEMENT		-
78	supervisor	ELEMENT		-
79	surname	ELEMENT		-
80	termination-date	ELEMENT		-
81	total-budget	ELEMENT		-
82	total-deductions	ELEMENT		-
83	total-hours	ELEMENT		-
84	zip-code	ELEMENT		-
85	department-information	ENTITY	dept-info	
86	hourly-employee-information	ENTITY	h-emp-info	
87	salaried-employee-information	ENTITY	s-emp-info	
88	h-gross-pay-comp-init	EVENT		-
89	h-report-entry-complete	EVENT		-
90	hourly-emp-processing-init	EVENT		-
91	init-hourly-paycheck-procedure	EVENT		-
92	new-employee-processing-init	EVENT		-
93	passed-error-checks	EVENT		-
94	salaried-emp-processing-init	EVENT		-
95	termination-processing-init	EVENT		-
96	time-card-found	EVENT		-
97	time-card-missing	EVENT		-
98	time-cards-collected	EVENT		-
99	total-hours-comp-init	EVENT		-
100	validity-check	EVENT		-
101	address	GROUP		-
102	check	GROUP		-
103	department-update-data	GROUP		-
104	emp-termination-data	GROUP		-
105	employee-name	GROUP		-
106	error-listing-entry	GROUP		-
107	h-derived-pay-data	GROUP		-
108	h-emp-report-entry	GROUP		-
109	hired-report-entry	GROUP		-
110	hourly-emp-pay-data	GROUP		-
111	hourly-job-data	GROUP		-
112	pay-stub	GROUP		-

1696

FIGURE_41

NAME LIST

	NAME	TYPE	SYNONYM	
113	personal-data	GROUP		-
114	s-derived-pay-data	GROUP		-
115	s-emp-report-entry	GROUP		-
116	salaried-emp-pay-data	GROUP		-
117	salaried-job-data	GROUP		-
118	term-report-entry	GROUP		-
119	time-card-data	GROUP		-
120	employee-information	INPUT	emp-info i1	
121	employment-termination-form	INPUT	term-form	
122	hourly-employment-form	INPUT	h-emp-form	
123	salaried-employment-form	INPUT	s-emp-form	
124	tax-withholding-certificate	INPUT	tax-cert	
125	time-card	INPUT	t-card	
126	departments-and-employees	INTERFACE	dept-emp r1	
127	employee	INTERFACE	emp	
128	payroll-department	INTERFACE	pay-dept	
129	day	INTERVAL		-
130	month	INTERVAL		-
131	week	INTERVAL		-
132	year	INTERVAL		-
133	electronic	KEYWORD		-
134	independent	KEYWORD		-
135	manual	KEYWORD		-
136	terminal	KEYWORD		-
137	test-object	KEYWORD		-
138	isdos-project-ann-arbor	MAILBOX		-
139	goals-memo	MEMO		-
140	i-o-constraints-memo	MEMO		-
141	process-memo	MEMO		-
142	processor-memo	MEMO		-
143	resource-memo	MEMO		-
144	rup-memo	MEMO		-
145	unit-memo	MEMO		-
146	error-listing	OUTPUT	e-list	
147	hired-employee-report	OUTPUT	hired-report	
148	hourly-employee-report	OUTPUT	h-emp-report	
149	pay-statement	OUTPUT	paycheck	

FIGURE_41

NAME LIST

	NAME	TYPE	SYNONYM
150	paysystem-outputs	OUTPUT	p1 payouts
151	salaried-employee-report	OUTPUT	s-emp-report
152	terminated-employee-report	OUTPUT	term-report
153	(	PROBLEM-DEFINER	-
154	(_><(/.-+)	PROBLEM-DEFINER	-
155	!	PROBLEM-DEFINER	-
156	$	PROBLEM-DEFINER	-
157	)	PROBLEM-DEFINER	-
158	%	PROBLEM-DEFINER	-
159	#	PROBLEM-DEFINER	-
160	@	PROBLEM-DEFINER	-
161	james-m-aaster	PROBLEM-DEFINER	-
162	michel-j-bastarache	PROBLEM-DEFINER	-
163		PROBLEM-DEFINER	-
164		PROBLEM-DEFINER	-
165		PROBLEM-DEFINER	-
166	department-file-addition	PROCESS	-
167	department-file-removal	PROCESS	-
168	federal-deductions-update	PROCESS	-
169	fica-deductions-update	PROCESS	-
170	funds-update	PROCESS	-
171	gross-pay-update	PROCESS	-
172	h-gross-pay-computation	PROCESS	-
173	h-report-entry-generation	PROCESS	-
174	hire-report-entry-generation	PROCESS	-
175	hourly-emp-processing	PROCESS	-
176	hourly-emp-update	PROCESS	-
177	hourly-employee-processing	PROCESS	-
178	hourly-information-creation	PROCESS	-
179	hourly-information-deletion	PROCESS	-
180	hourly-paycheck-production	PROCESS	-
181	hourly-paycheck-validation	PROCESS	-
182	hours-update	PROCESS	-
183	net-pay-computation	PROCESS	-
184	new-employee-processing	PROCESS	-
185	pay-computation-validation	PROCESS	-
186	payroll-processing	PROCESS	payproc p1

FIGURE_41

NAME LIST

	NAME	TYPE	SYNONYM	
187	process-library	PROCESS		-
188	s-gross-pay-computation	PROCESS		-
189	s-report-entry-generation	PROCESS		-
190	salaried-emp-update	PROCESS		-
191	salaried-employee-processing	PROCESS		-
192	salaried-information-creation	PROCESS		-
193	salaried-information-deletion	PROCESS		-
194	salaried-paycheck-production	PROCESS		-
195	salaried-paycheck-validation	PROCESS		-
196	state-deductions-update	PROCESS		-
197	tax-computation	PROCESS		-
198	term-report-entry-generation	PROCESS		-
199	terminating-emp-processing	PROCESS		-
200	time-card-validation	PROCESS		-
201	total-deductions-computation	PROCESS		-
202	total-hours-computation	PROCESS		-
203	computer-processor	PROCESSOR		-
204	payroll-processor	PROCESSOR		-
205	validation-clerk	PROCESSOR	valid-clerk	
206	dept-hourly-emp-relation	RELATION		-
207	dept-salaried-emp-relation	RELATION		-
208	cpu-time	RESOURCE		-
209	paper	RESOURCE	pr	
210	job	RESOURCE-USAGE-PARAMETER		-
211	payroll	RESOURCE-USAGE-PARAMETER		-
212	validation	RESOURCE-USAGE-PARAMETER	valid	
213	company-only	SECURITY		-
214	no-restrictions	SECURITY		-
215	unclassified	SECURITY		-
216	department-file	SET	dept-file	
217	hourly-employee-file	SET	h-emp-file	
218	payroll-master-information	SET	master-file pay-mast s1	
219	salaried-employee-file	SET	s-emp-file	
220	constantine	SOURCE		-
221	processor-specification	SOURCE		-
222	resource-specification	SOURCE		-
223	rup-specification	SOURCE		-

169f

FIGURE_41

NAME LIST

	NAME	TYPE	SYNONYM	
224	unit-specification	SOURCE		-
225	many	SYSTEM-PARAMETER		-
226	no-hourly-emp-processing	SYSTEM-PARAMETER		-
227	no-new-emp-processing	SYSTEM-PARAMETER		-
228	no-of-departments	SYSTEM-PARAMETER		-
229	no-of-employees	SYSTEM-PARAMETER		-
230	no-of-hourly-employees	SYSTEM-PARAMETER		-
231	no-of-payroll-processing	SYSTEM-PARAMETER		-
232	no-of-salaried-employees	SYSTEM-PARAMETER		-
233	no-of-supervisors	SYSTEM-PARAMETER		-
234	no-salaried-emp-processing	SYSTEM-PARAMETER		-
235	no-terminating-emp-processing	SYSTEM-PARAMETER		-
236	several	SYSTEM-PARAMETER		-
237	page	UNIT	pg	
238	seconds	UNIT		-

PICTURE

Purpose

The PICTURE report presents flow and structure information about the problem statement in a graphical format. The PICTURE report provides the user with a detailed view of one part of the target system description (i.e., it presents all flow and structure relationships a particular name has with other names).

Information Presented

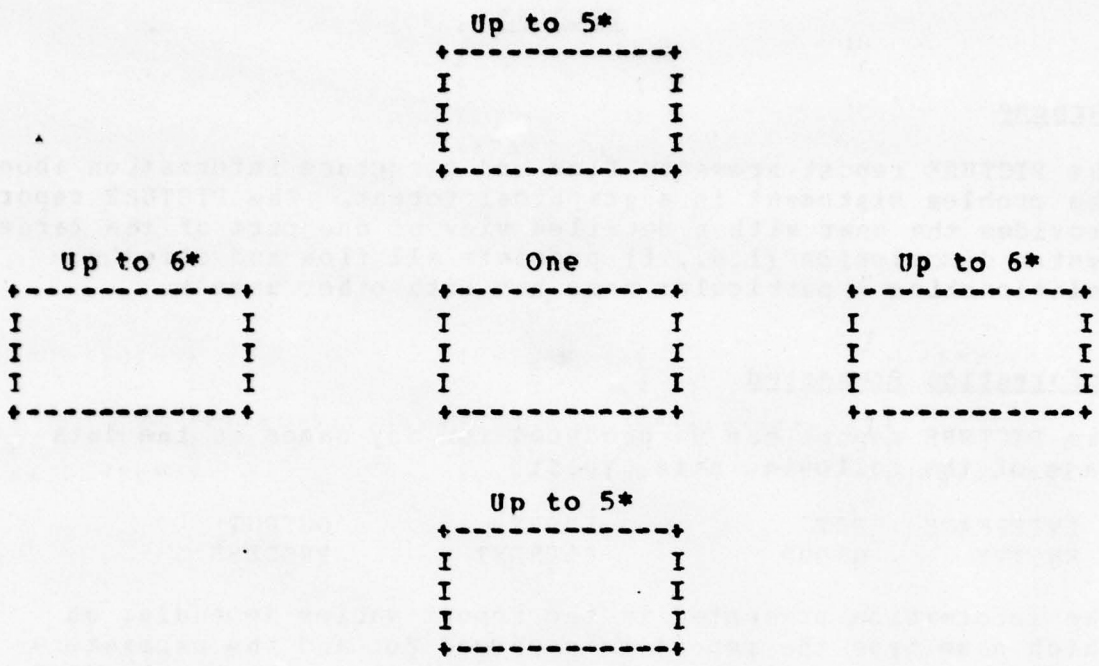
The PICTURE report can be produced for any names in the data base of the following name types:

INTERFACE	SET	INPUT	OUTPUT
ENTITY	GROUP	ELEMENT	PROCESS

The information presented in the report varies depending on which name type the report is produced for and the parameters used when generating the report. For each name type the FLOW, STRUCTURE and DATA parameters present different types of URL relationships as retrieved from the data base. Table 7 shows the relationships presented for each name type.

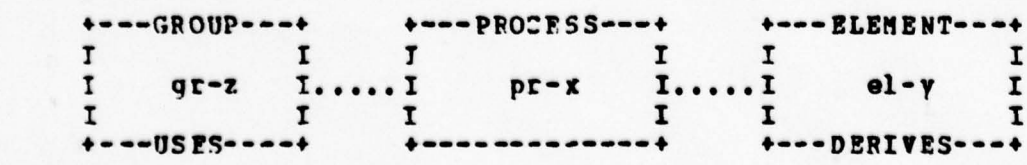
Format

The PICTURE report is generated in a graphical format. The basic template for the format is shown in Figure 42. A given PICTURE report describes a single named object.



General PICTURE Format and Limits per Page
Figure 42

The object being described is represented by a retangular box printed in the center of the report page. All named objects related to the center object are also represented by rectangular boxes but are arranged around the perimeter of the page. The name type (PROCESS, ELEMENT, etc.) of the represented object is printed along the top line of each box. The relationship of an object (represented by one of the perimeter boxes) with the center object is printed along the bottom line of the box. For illustrative purposes, lines extend from each box to the center box.

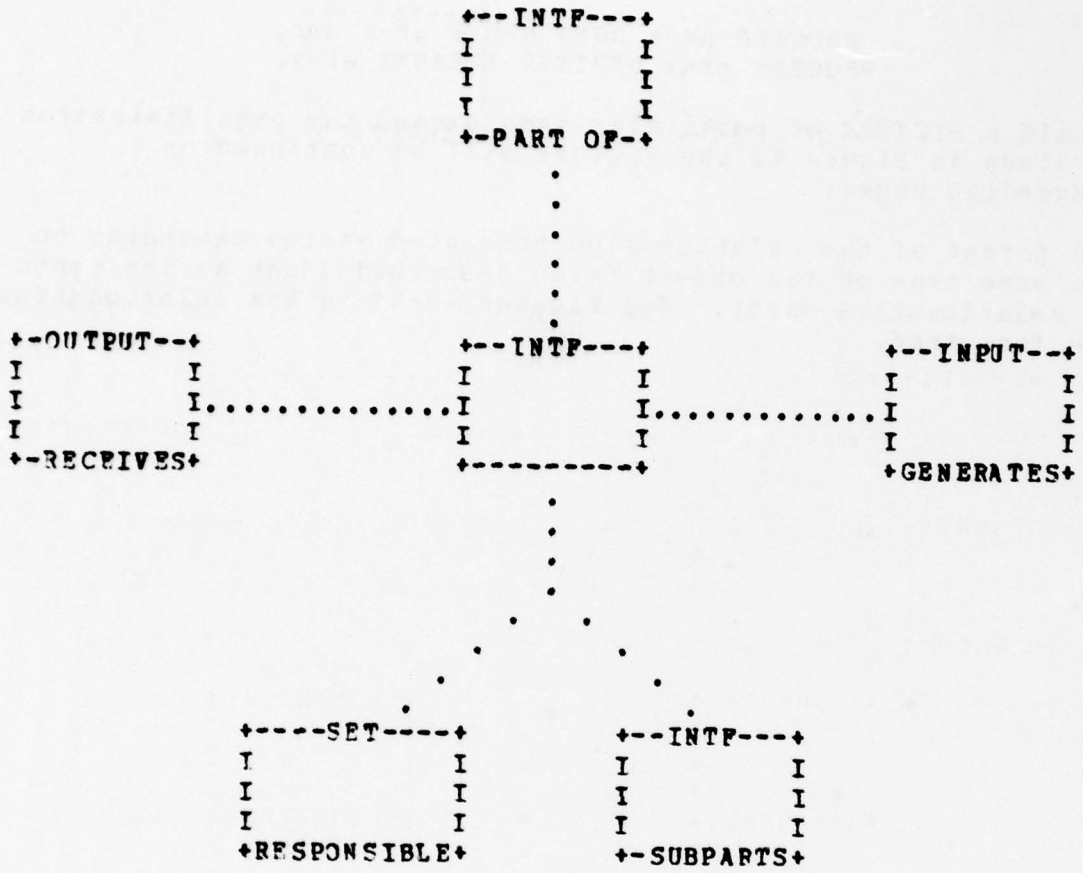


The preceding example is to be interpreted as follows:

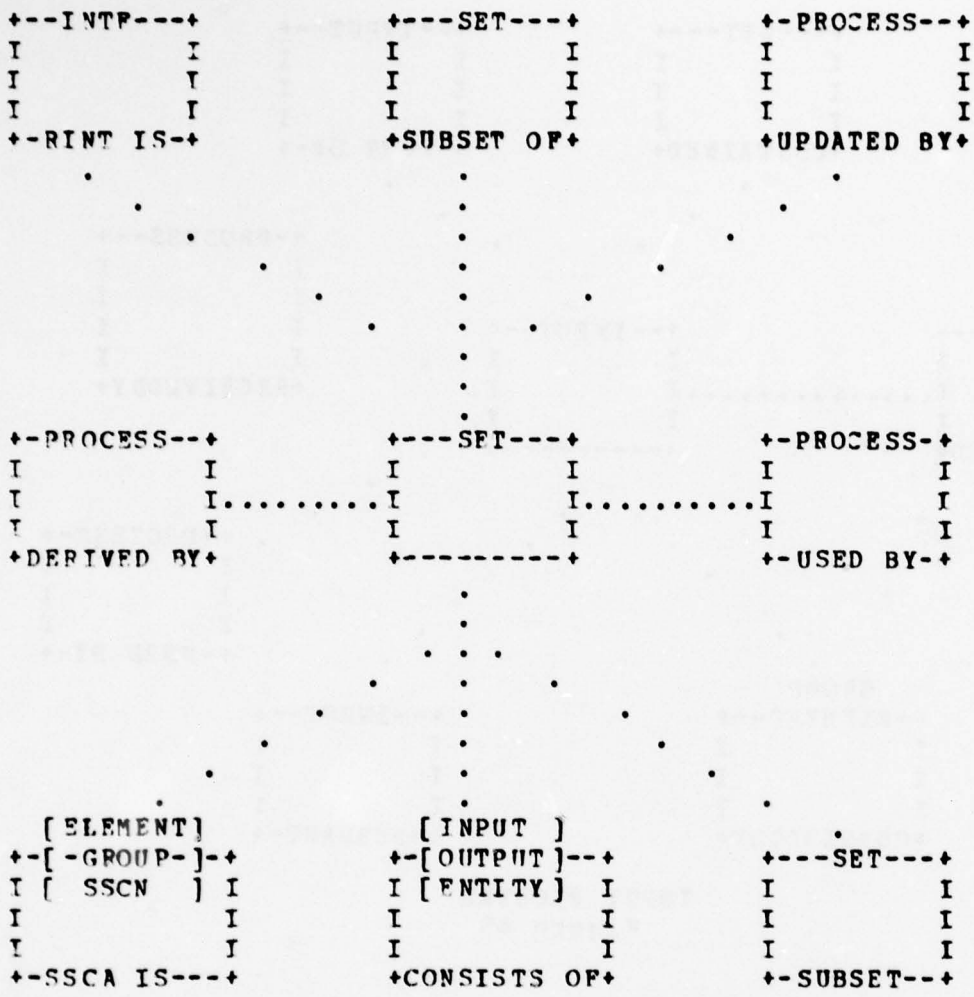
PROCESS pr-x USES GROUP gr-z and,
PROCESS pr-x DERIVES ELEMENT el-y.

Should a PICTURE of particular name exceed the page limitation as given in Figure 42 the PICTURE will be continued on succeeding pages.

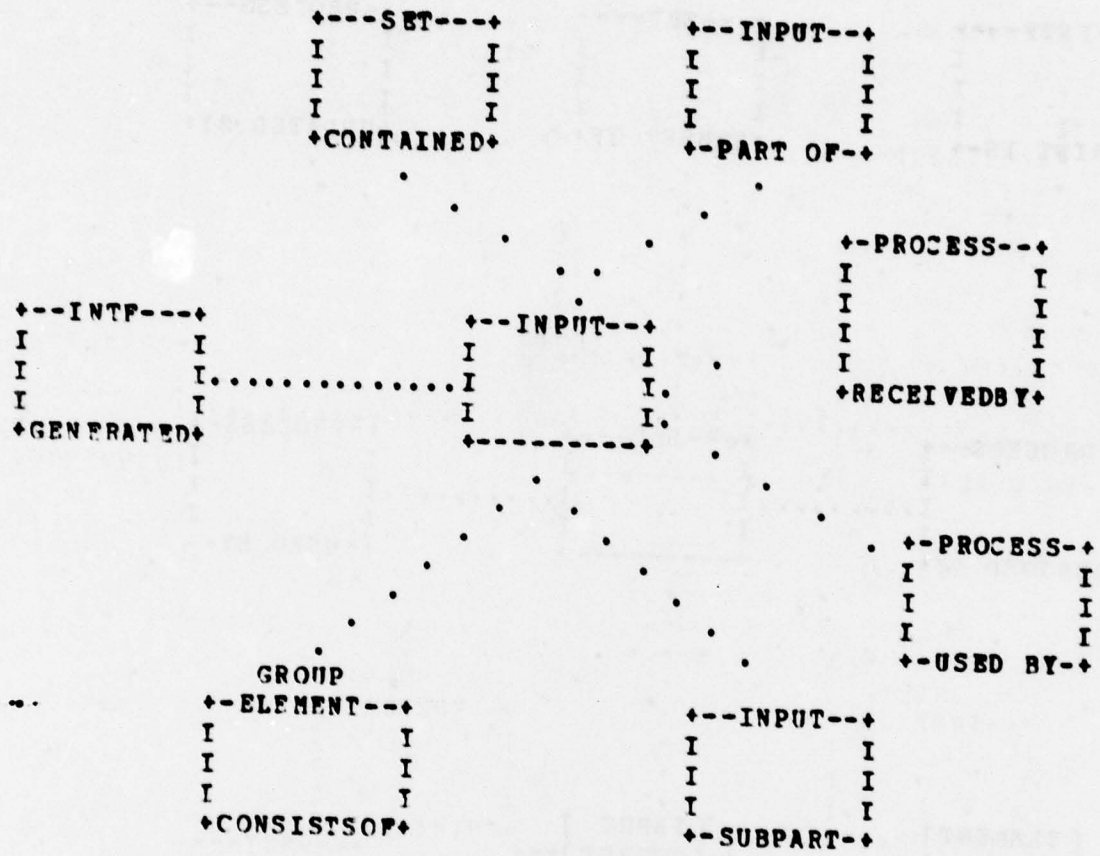
The format of the relationships presented varies depending on the name type of the object being described (just as the types of relationships vary). See Figures 43-49 on how relationships are formatted.



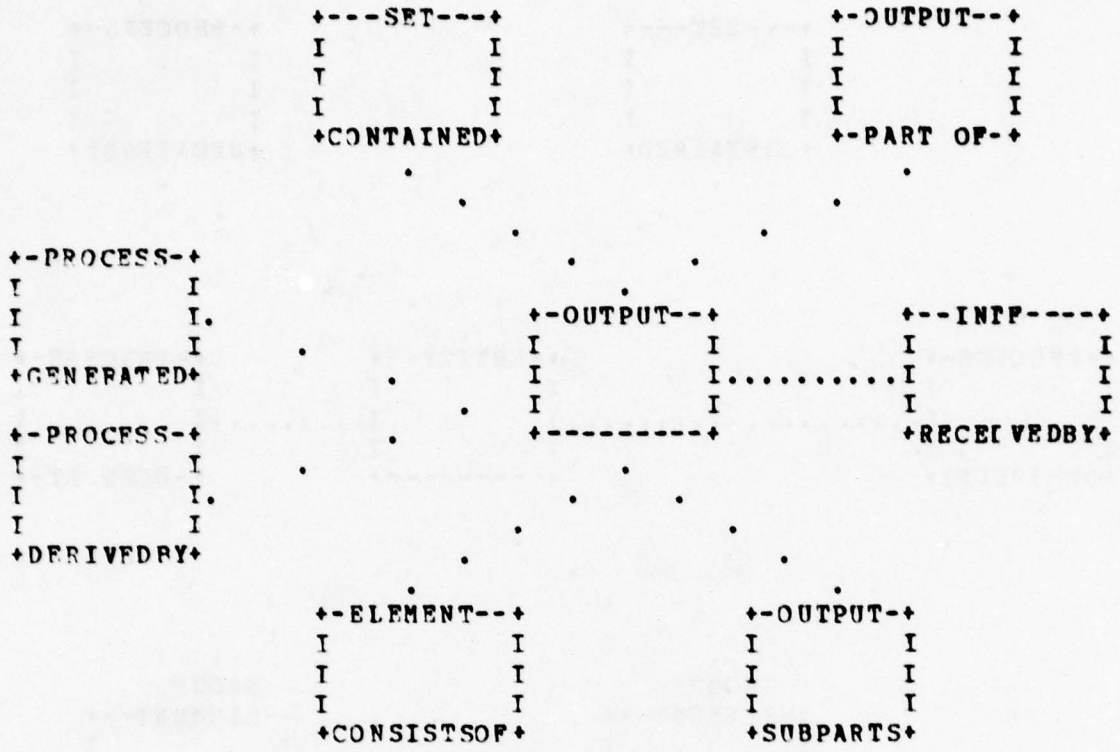
INTERFACE PICTURE
Figure 43



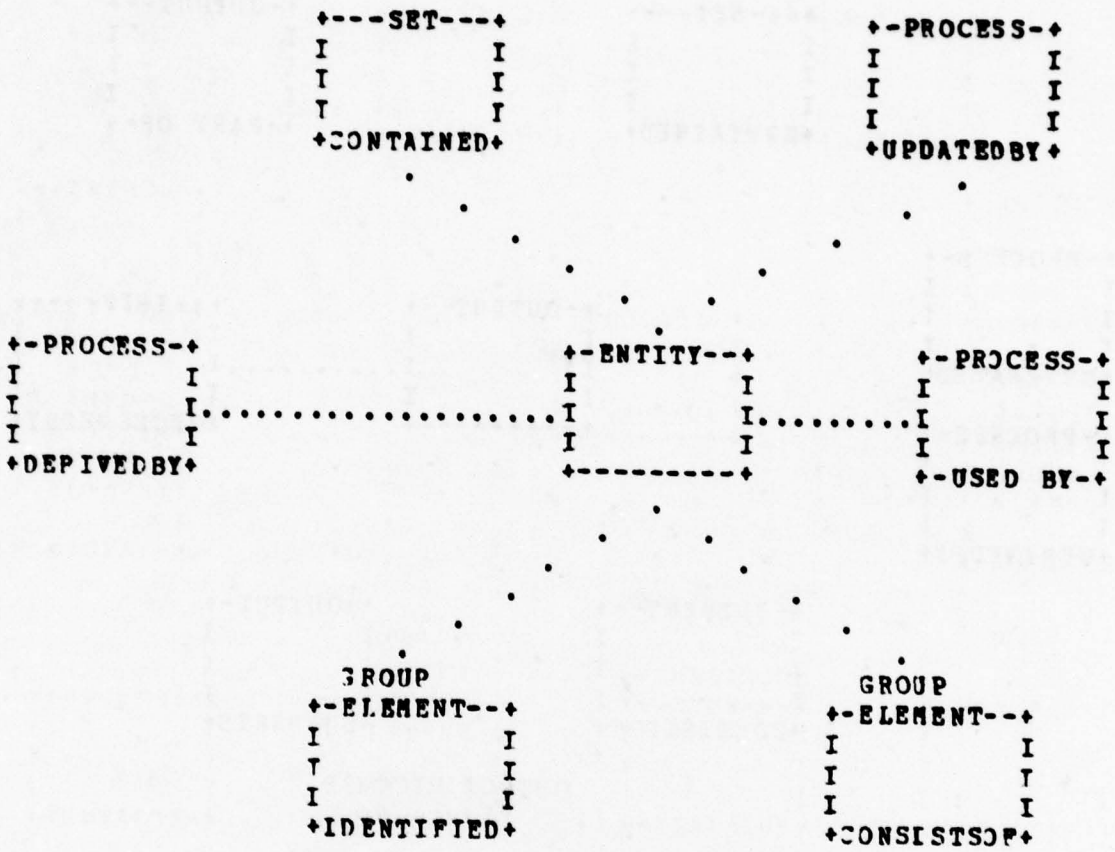
SET PICTURE
Figure 44



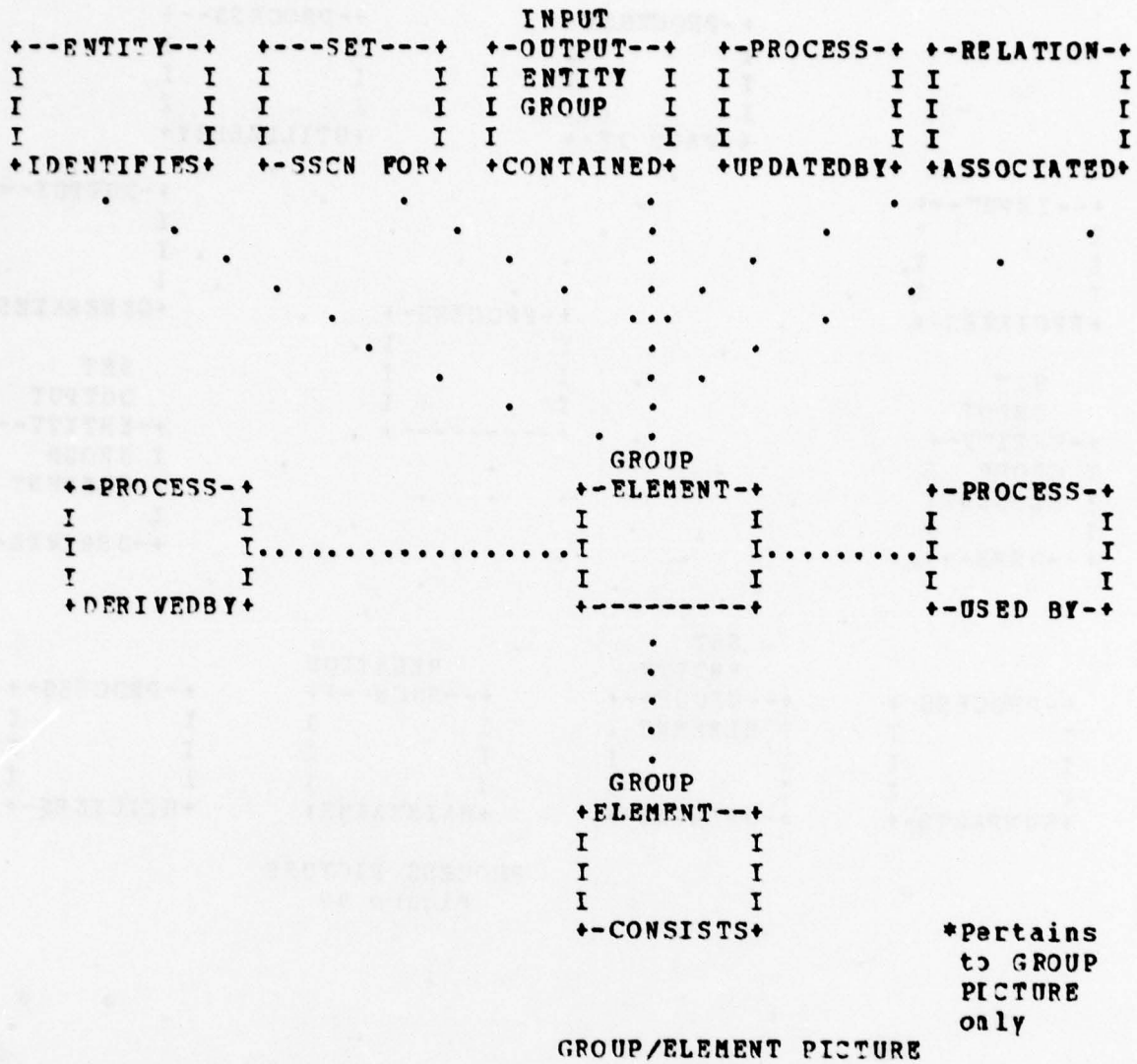
INPUT PICUTRE
Figure 45



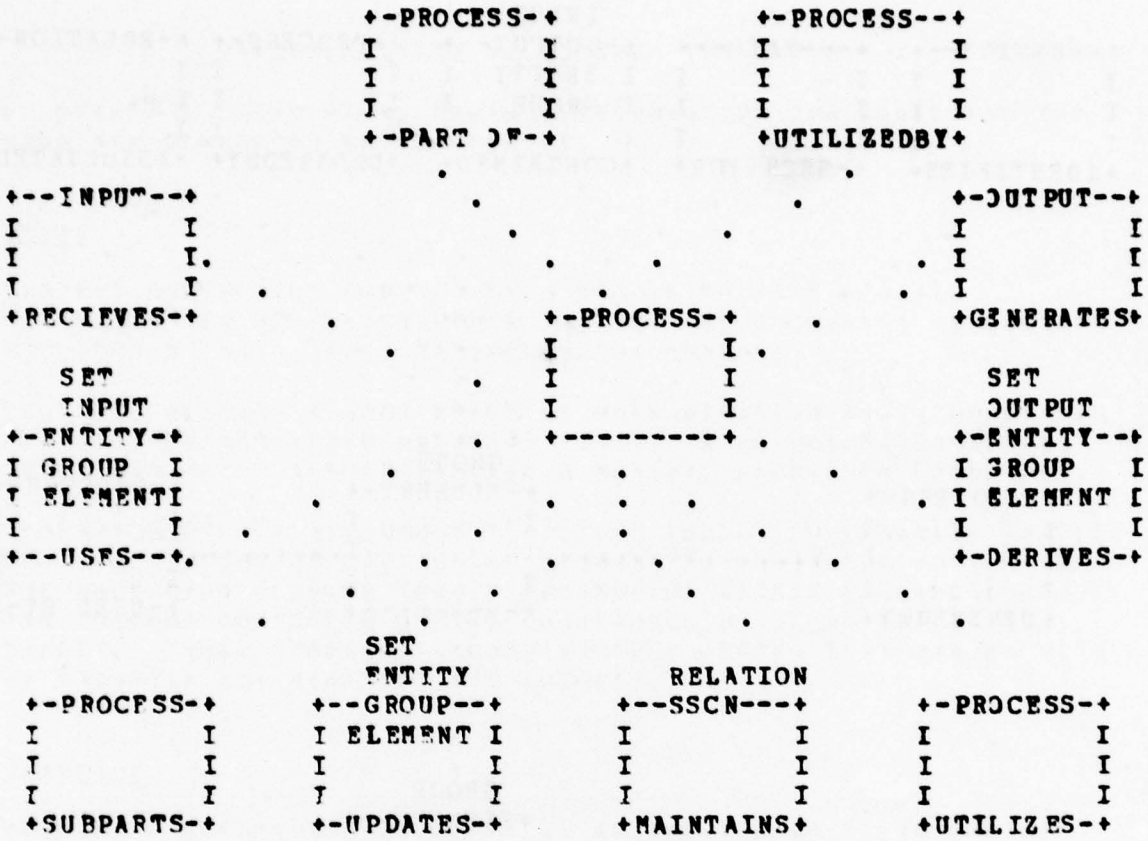
OUTPUT PICTURE
Figure 46



ENTITY PICTURE
Figure 47



GROUP/ELEMENT PICTURE
Figure 48



PROCESS PICTURE
Figure 49

Options and Alternatives

The user has the option of displaying any combination of the three types of relationships (DATA, FLOW and STRUCTURE) on the report. See Table 7 for which relationships are displayed in each of these categories for a particular name type. For example, if only the CONSISTS and CONTAINED information (STRUCTURE) were to be displayed for a particular ENTITY name, DATA and FLOW relationships could be suppressed via the NODATA and NOFLOW parameters. The parameters allowed and their effect on the report are described below:

1) DATA - specifies that data type relationships be included in each PICTURE.
   NODATA - specifies that these relationships are not included.

2) FLOW - specifies that flow type relationships be included in each PICTURE.
   NOFLOW - specifies that these relationships are not included.

3) STRUCTURE - specifies that structure type relationships be included in each PICTURE.
   NOSTRUCTURE - specifies that these relationships are not included.

An INDEX for the report is produced when the INDEX parameter is used.

The report may be generated for a single input name (via the NAME parameter) or for a collection of names either specified by the USER or retrieved via NAME-GEN.

Analysis

For each name given as input the software finds the name in the data base. If the name is not found the message:

URA066: MAINPIC: NAME NOT IN D.B. -

is printed. If the name is found it is checked if it is of a legal name type for which a PICTURE may be generated, i.e., a SET, INPUT, OUTPUT, ENTITY, GROUP, ELEMENT, PROCESS, or INTERFACE name. If it is not one of these name types the message:

URA067:MAINPIC: PICTURE NOT AVAILABLE FOR -

is printed. If the name is of a legal name type it is then checked if any of the relationships that can be presented in the PICTURE for that name type exist for the particular name. If none of these relationships exist, the message:

URA289:PCLRBT: NO PICTURE AVAILABLE FOR

is printed. Otherwise, those relationships available for the name are presented in the report.

Usage

Project management can use this report to gain a basic understanding of the functions of the target system by viewing PICTURES of high level target system objects.

PICTURES provide a good means of communication among people in the project and those external to it. A graphical format is often easier to interpret than a matrix, narrative text, etc.

Problem Definers may use the PICTURE report to visually analyze the description of particular objects to check for completeness. For each type of name (e.g., PROCESS or ELEMENT) several checks can be made depending on the relationships presented in the report. Table 8 presents completeness checks that can be made by visually scanning PICTURE reports.

Examples

Figure 50 presents a PICTURE for an INPUT name "time-card."

Figure 51 presents a PICTURE for an INTERFACE name "employee."

Figure 52 presents a PICTURE for a PROCESS name "hourly-employee-processing." This example was produced by giving the following Analyzer command:

```
PICTURE NAME=hourly-employee-processing
```

FIGURE_50

PICTURE

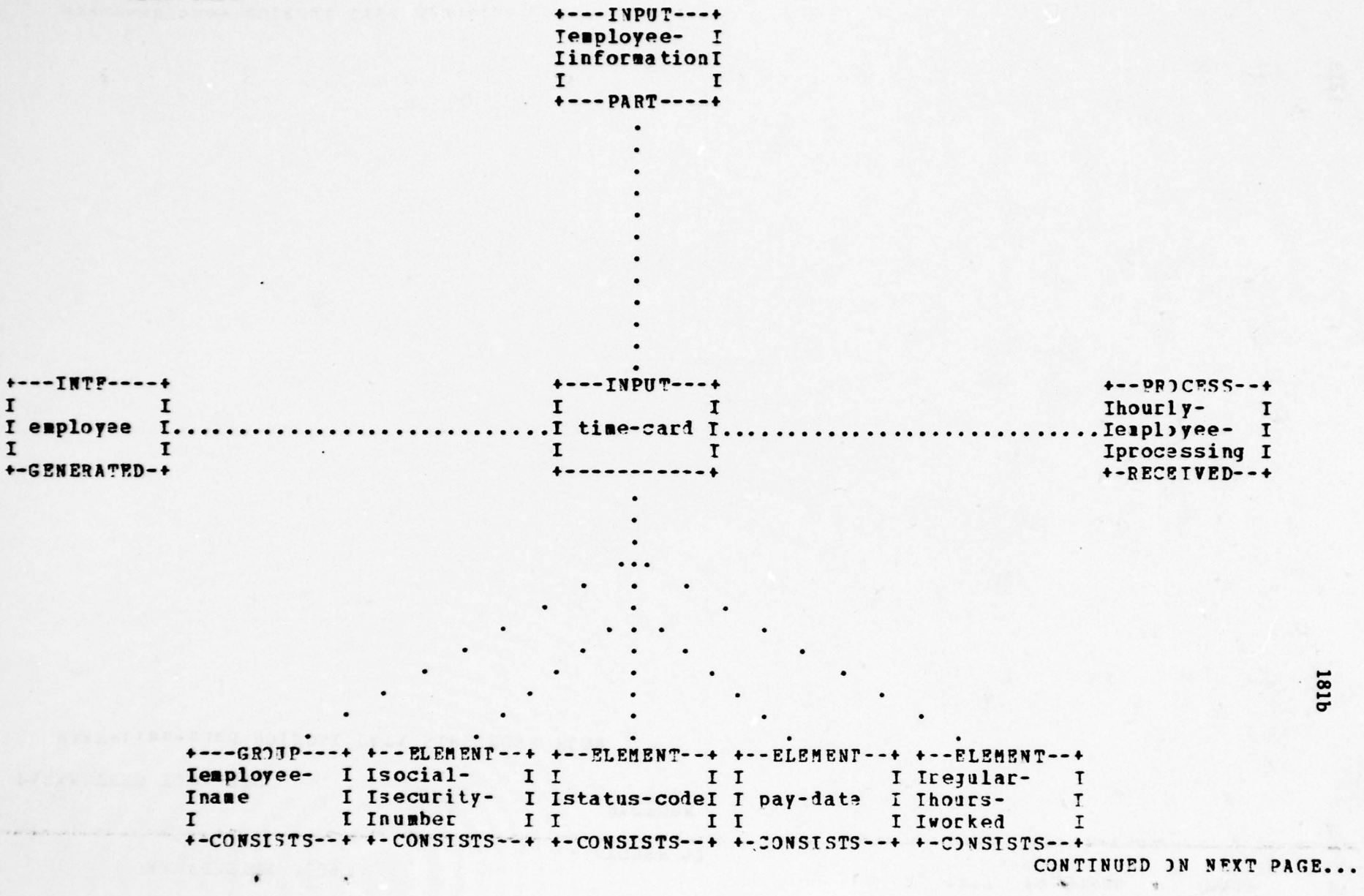
PARAMETERS FOR: PIC

NAME=time-card NOINDEX DATA STRUCTURE FLOW

FIGURE_50

INPUT PICTURE

time-card



CONTINUED ON NEXT PAGE...

FIGURE_50

INPUT PICTURE

time-card (CONTINUED)

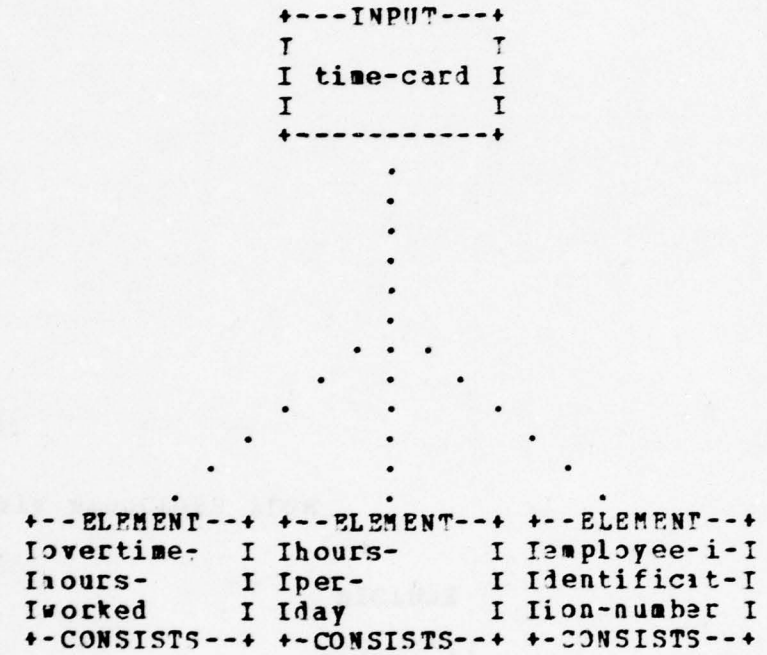


FIGURE_51

PICTURE

PARAMETERS FOR: PIC

NAME=employee NOINDEX DATA STRUCTURE FLOW

FIGURE_51

INTERFACE PICTURE

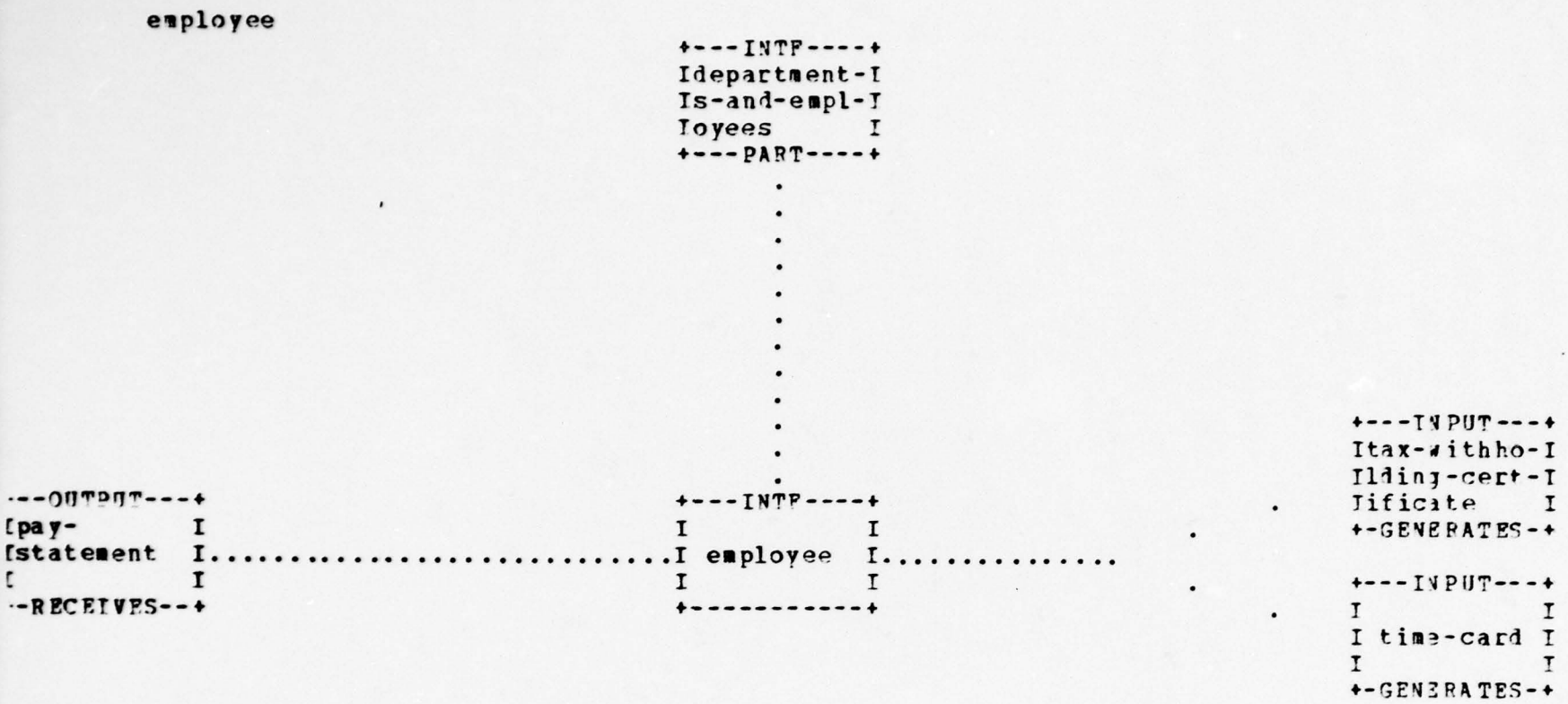


FIGURE_52

PICTURE

PARAMETERS FOR: PIC

NAME=hourly-employee-processing NOINDEX DATA STRUCTURE FLOW

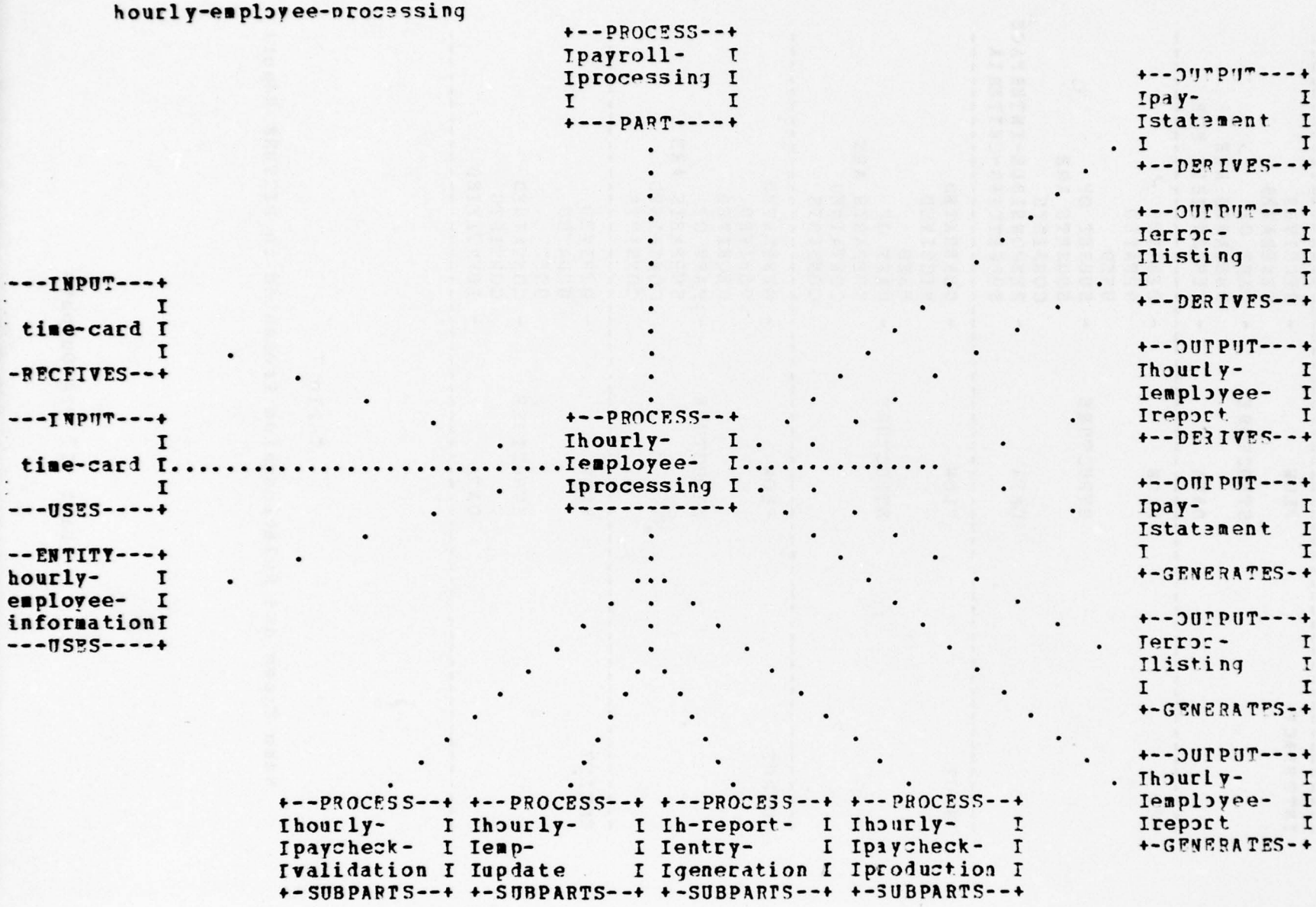
AD-A057 130 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS ANALYZER (URA) USER'S MANUAL H6180/MULTICS/VE--ETC(U)
MAR 77 D TEICHROEW, A HERSHEY, S SPEWAK F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-128 NL

4 OF 6
AD A057130



FIGURE_52

PROCESS PICTURE



NAME TYPE		RELATIONSHIPS DISPLAYED
INTERFACE	FLOW	- RECEIVES GENERATES
	STRUCTURE	- PART OF SUBPARTS ARE
	DATA	- RESPONSIBLE FOR
SET	FLOW	- DERIVED UPDATED USED
	STRUCTURE	- SUBSET OF SUBSETS ARE CONSISTS
	DATA	- RESPONSIBLE-INTERFACE SUBSETTING-CRITERIA
INPUT	FLOW	- GENERATED RECEIVED USED
	STRUCTURE	- PART OF SUBPARTS ARE CONTAINED CONSISTS
OUTPUT	FLOW	- GENERATED DERIVED RECEIVED
	STRUCTURE	- PART OF SUBPARTS ARE CONTAINED CONSISTS
ENTITY	FLOW	- DERIVED UPDATED USED
	STRUCTURE	- CONTAINED CONSISTS
	DATA	- IDENTIFIED

Table 7

Name Types and Relationships Presented in PICTURE Report

NAME TYPE		RELATIONSHIPS DISPLAYED
GROUP/ELEMENT	FLOW	- DERIVED
		UPDATED
		USED
		USED TO DERIVE
		USED TO UPDATE
	STRUCTURE	- CONTAINED
		CONSISTS[1]
	DATA	- ASSOCIATED
		IDENTIFIED
		SUBSETTING-CRITERION
PROCESS	FLOW	- RECEIVES
		USES
		USES TO DERIVE
		USES TO UPDATE
		DERIVES
		GENERATES
		UPDATES
		MAINTAINS
	STRUCTURE	- PART OF
		SUBPARTS ARE
		UTILIZED BY
		UTILIZES

Table 7 (Continued)

---

[1] This relationship only applies to GROUPS, i.e., an ELEMENT cannot CONSIST of anything.

INTERFACE	An INTERFACE should RECEIVE an OUTPUT, GENERATE an INPUT and/or be RESPONSIBLE for a SET.
SET	A SET should be USED by a PROCESS, DERIVED by a PROCESS, and/or be UPDATED by a PROCESS. A check can also be made that the SET has a RESPONSIBLE-INTEFACE. If the SET has SUBSETS, it should also have SUBSETTING-CRITERIA.
INPUT	An INPUT should be RECEIVED by a PROCESS and GENERATED by an INTERFACE. An INPUT should also be DERIVED by a PROCESS.
OUTPUT	An OUTPUT should be GENERATED by a PROCESS and RECEIVED by an INTERFACE. An OUTPUT should also be DERIVED by a PROCESS.
ENTITY	An ENTITY should be USED by a PROCESS, DERIVED by a PROCESS, and/or be UPDATED by a PROCESS. A check can also be made that the ENTITY is IDENTIFIED by a GROUP or ELEMENT.
GROUP/ELEMENT	A GROUP/ELEMENT should be USED by a PROCESS, DERIVED by a PROCESS, and/or be UPDATED by a PROCESS. A check can be made that the GROUP/ELEMENT may IDENTIFY an ENTITY, be SUBSETTING-CRITERION for a SET and/or be ASSOCIATED with a RELATION.
PROCESS	A PROCESS should RECEIVE an INPUT, GENERATE an OUTPUT, USE a SET, ENTITY, INPUT, GROUP or ELEMENT, DERIVE a SET, OUTPUT, ENTITY, GROUP or ELEMENT, UPDATE a SET, ENTITY, GROUP or ELEMENT, and/or MAINTAIN a RELATION

Table 8

Completeness Checks that may be made
by the PICTURE Report.

PROCESS CHAIN REPORT

Purpose

To present in a graphical format the sequence of EVENTS and PROCESSES which occurs as a result of each EVENT or PROCESS specified as input.

Information Presented

For each EVENT name given as input to the software producing the report, the report presents all PROCESS names which the EVENT TRIGGERS. For each of these PROCESS names presented the report presents all EVENTS occurring ON-INCEPTION or ON-TERMINATION of the PROCESS. The report then presents all the PROCESSES which these EVENTS TRIGGER, etc., and the network continues until a loop is encountered or no more relationships are found.

For each PROCESS name given as input to the software producing the report, the report presents all EVENTS occurring ON-INCEPTION or ON-TERMINATION of the PROCESS. For each of these EVENT names presented the report presents all PROCESS names which the EVENT TRIGGERS. The report then presents all the EVENTS occurring ON-INCEPTION or ON-TERMINATION of the PROCESS, etc., and the network continues until a loop is encountered or no more relationships are found.

Format

Each name which appears on the output is shown within a box. The top line of the box indicates the name type (EVENT or PROCESS), while the bottom line shows the relationship with the preceding EVENT or PROCESS (TRIGGERED, ON-INCEPTION, ON-TERMINATION). Boxes containing related names are linked by dotted lines.

If a name joins two or more chains (strings of related names) into a loop or loops, every appearance of that name after the first will be linked to a box containing the message, "LOOPS TO PREVIOUS ENTRY."

Output is continued across page boundaries. If the right edge of one page continues to the left edge of a second, the right most column of boxes on the first page will be repeated as the left most column of boxes on the second page, in order to facilitate matching of edges. Similarly, if the bottom edge of one page continues to the top edge of a second, the bottom row of boxes on the first page will be repeated as the top row of boxes on the second page.

Options and Alternatives

The report may be generated for a single EVENT or PROCESS name (via the NAME parameter) or for a collection of such names, either input by the user or obtained by use of NAME-GEN.

The number of columns and rows used on the page may be decreased from their default values of 119 and 39, respectively, via the COLUMNS and ROWS parameters. The minimum acceptable values for COLUMNS and ROWS are 38 and 14, respectively.

The number of boxes arranged horizontally or vertically on a page may be decreased from the defaults, which are the maximum numbers that will fit in each direction (depending on COLUMNS and ROWS), in order to make the output less cluttered. The parameters which can be used to do this are HORIZONTAL-BOXES and VERTICAL-BOXES. Their maximum values (for COLUMNS=119 and ROWS=39) are 6. Due to the scheme for continuing pages, their minimum values are 2.

The number of connections to be traced, starting at the given name, may be set at any positive value via the LINKS parameter. The same LINKS value is used for all names input when the FILE parameter is used.

An index, containing each name used on the report and the page(s) on which it appears, may be obtained by specifying the INDEX parameter.

Analysis

Each name given as input is first checked to see that it is in the data base and that it is either a PROCESS or EVENT name. If the name is not in the data base, the message:

URA360: FILMAT: NAME NOT IN DATA BASE

will be given. If the name is of a type other than PROCESS or EVENT, the user will receive the message:

URA361: FILMAT: NAME NOT EVENT OR PROCESS.

If the name passes these two tests, it is placed in a data structure which will later be used for ouput. The name is then used to generate a tree structure of PROCESSES and EVENTS as follows. If the name is an EVENT, all PROCESSES which it TRIGGERS are retrieved from the data base and placed on a stack. Then, the first name is removed from the stack, placed in its proper location in the data structure, and all EVENTS occurring ON-INCEPTION or ON-TERMINATION of that PROCESS are retrieved and placed on the stack. This procedure continues, with names being removed from the top of the stack, placed in the data structure, and used to obtain further names, which are then placed on the

stack. At any stage of this procedure, no names will be put on the stack if one of the following is true:

1) The current name is an EVENT which TRIGGERS no PROCESSES, or a PROCESS which has no EVENTS ocurring ON-INCEPTION or ON-TERMINATION.

2) The current name has been encountered earlier, and is therefore at the end of a chain or forms a loop with some portion of a chain traced earlier.

3) The number of links that has been traced on the current chain is equal to the limit set by the LINKS parameter. (For every input name for which this occurs, the message:

URA364: FILMAT: LINK LIMIT SPECIFIED WAS REACHED

will be printed.)

Thus, in any of these cases, the size of the stack will decrease. The entire procedure is complete when, after any search, the stack is empty. If the name input is a PROCESS name, the first search is for EVENTS occurring ON-INCEPTION or ON-TERMINATION of that PROCESS. From there, the procedure is identical to that described above.

The data structure constructed from all names found as above is broken into page-size units and is printed a page at a time.

The process described above is repeated until no more names remain in the input stream.

Usage

The PROCESS CHAIN report presents a comprehensive view of the dynamic behavior of the PROCESSES within the target system for inclusion in the final specifications of the system or as an aid in communicating this information to others.

Programmers and System Designers in particular will find this report helpful in identifying and optimizing the system logic. If the PROCESSES are defined to the level of computable statements, the PROCESS CHAIN report will essentially chart out the program logic.

Problem Definers may use the PROCESS CHAIN report to visually analyze the description of particular objects and the system as a whole, for completeness. Table 9 presents completeness checks that can be made by visually scanning PROCESS CHAIN reports.

Example

Figure 53 presents a PROCESS CHAIN for the EVENT "salaried-emp-processing-init."

FIGURE_53

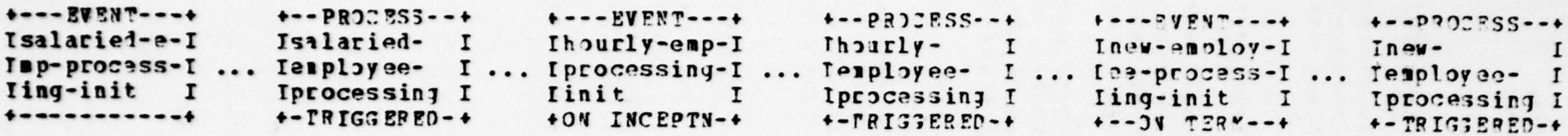
Process Chain

PARAMETERS FOR: PC

NAME=salaried-emp-processing-init LINKS=1000 NOIND3X COLUMNS=110 ROWS=30 HORIZONTAL-BOXES=6
VERTICAL-BOXES=6

FIGURE_53

Process Chain



PROCESS	The absence of any EVENT as a result of INCEPTION or TERMINATION of the PROCESS should be rationalized.
EVENT	The absence of any PROCESS being TRIGGERED by an EVENT should be rationalized.
System Description	All chains should terminate in one of three ways: 1) In a loop back into the chain 2) By a PROCESS designating the last activity in the procedure represented by the chain 3) By an EVENT designating termination of a procedure represented by a chain Given a particular PROCESS or EVENT, the report allows a trace to be made through the system of actions taken. Checks can be made that based on a particular starting point all EVENT-PROCESS chains evolving from the starting point terminate correctly.

Table 9

Completeness Checks that may be made by
Visual Analysis of the PROCESS CHAIN Report

PROCESS INPUT/OUTPUT

Purpose

To present in an easy to examine outline form, the basic functions of one or more PROCESSES in the Language description and how these PROCESSES interact with information.

Information Presented

The report presents, for the PROCESS names given as input, four types of information which may be printed or suppressed by the specification of appropriate parameters for the command generating the report.

The DESCRIPTION parameter permits the printing of the DESCRIPTION comment entry for each PROCESS if available. The PROCEDURE parameter permits the printing of the PROCEDURE comment entry for each PROCESS if available.

The INPUT parameter permits the printing of the names of all SETS, INPUTS, ENTITIES, GROUPS and/or ELEMENTS that are RECEIVED and/or USED by each PROCESS. The OUTPUT parameter permits the printing of the names of all SETS, OUTPUTS, ENTITIES, GROUPS and/or ELEMENTS that GENERATED, UPDATED and/or DERIVED by each PROCESS.

Format

An entry in the report is printed for each PROCESS name given as input. Each name is identified by a number, 1*, 2*, etc., designating its position in the input stream. The following format is used to print out information about each process:

```
**   process name

        [DESCRIPTION comment entry]
        ---------------------------
        [PROCEDURE comment entry]

           *** INPUTS ***

        [All INPUTS RECEIVED by the PROCESS]

        [All SETS, INPUTS, ENTITIES, GROUPS and ELEMENTS USED
        by the PROCESS]

           *** OUTPUTS ***

        [All OUTPUTS GENERATED by the PROCESS]
```

[All SETS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS DERIVED by the PROCESS]

[All SETS, ENTITIES, GROUPS and ELEMENTS UPDATED by the PROCESS]

All the names listed under the INPUTS and OUTPUTS headings are numbered sequentially.

If a DESCRIPTION or PROCEDURE comment entry is not available for a particular name, that part of the format is not included in the report. If no names are listed under the INPUTS heading, the message:

NO INPUTS FOR THIS PROCESS

will be printed. If no names are listed under the OUTPUTS heading, the message:

NO OUTPUTS FOR THIS PROCESS

will be printed.

Options and Alternatives

Any part of the information presented for each PROCESS name can be included in or omitted from the report depending on the parameters used when generated it. The parameters and their effect on the report are given below:

1) DESCRIPTION - specifies that the DESCRIPTION comment entry for each name be included in the report.

NODESCRIPTION - specifies that the comment entry is not printed.

2) PROCEDURE - specifies that the PROCEDURE comment entry for each name be included in the report.

NOPROCEDURE - specifies that the comment entry is not printed.

3) INPUT - specifies that names USED or RECEIVED by the PROCESS are presented in the report.

NOINPUT - specifies that those names are not printed.

4) OUTPUT - specifies that names DERIVED, UPDATED or GENERATED by the PROCESS are presented in the report.

NOOUTPUT - specifies that these names are not printed.

Each entry of the report is started at the beginning of a new page when the NEW-PAGE parameter is specified. When NONEW-PAGE is in effect, the entries are printed one after another in the report.

An INDEX for the report is produced when the INDEX parameter is specified.

The report may be generated for a single name (via the NAME parameter) or for a collection of names either specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input to the software producing the report is searched for in the data base. If it is not found the message:

URA076: MAINPRIO: NAME NOT IN DATA BASE -

is printed. If it is found a check is made that it is a PROCESS name. If it is not, the message:

URA088: MAINPRIO: NAME NOT A PROCESS NAME -

is printed. If the name is a PROCESS name, then the information available for it, as requested by the parameters, is presented on the report.

Usage

This report is beneficial in presenting a general description of the functions of target system (as described by the PROCESS) for purposes of communications between analysts and users.

It may also be used by analysts to check that the DESCRIPTION and PROCEDURE defined for each PROCESS is in agreement with the information which is input to or output from the PROCESS.

Examples

Figure 54 presents a PROCESS INPUT/OUTPUT report for a single name "payroll-processing."

Figure 55 presents the report generated for the SUBPARTS of "payroll-processing." This was done by the following commands:

```
NAME-GEN S='SUBPARTS-OF=payroll-processing,1'
```

PROCESS-INPUT-OUTPUT PROCEDURE

FIGURE_54

PROCESS INPUT/OUTPUT

PARAMETERS FOR: PRIO

NAME=payroll-processing INPUT OUTPUT DESCRIPTION NOPROCEDURE NONEW-PAGE NOINDEX PRINT NOPUNCH

1* payroll-processing

This process represents the highest level process
in the target system. it accepts and processes
all inputs and produces all outputs.

*** INPUTS ***

1	employee-information	RECEIVED
2	employee-information	USED
3	payroll-master-information	USED

*** OUTPUTS ***

1	paysystem-outputs	GENERATED
2	paysystem-outputs	DERIVED
3	payroll-master-information	UPDATED

FIGURE_55

PROCESS INPUT/OUTPUT

PARAMETERS FOR: PRIO

FILE INPUT OUTPUT DESCRIPTION PROCEDURE NONEW-PAGE NOINDEX PRINT NOPUNCH

1* hourly-employee-processing

This process performs those actions needed to interpret time cards to produce a pay statement for each hourly employee.

---

1. compute gross pay from time card .
2. compute tax from gross pay.
3. subtract tax from gross pay to obtain net pay.
4. update hourly employee information accordingly.
5. update department information accordingly.
6. generate paycheck.

Note: if status code specifies that the employee did not work this week, no processing will be done for this employee information.

*** INPUTS ***

1	time-card	RECEIVED
2	time-card	USED
3	hourly-employee-information	USED

*** OUTPUTS ***

1	pay-statement	GENERATED
2	error-listing	GENERATED
3	hourly-employee-report	GENERATED
4	pay-statement	DERIVED
5	error-listing	DERIVED
6	hourly-employee-report	DERIVED

..

FIGURE_55

PROCESS INPUT/OUTPUT

2* new-employee-processing

This process stores information about new employees and then prints out a corresponding report.

---

1. add new employee information
2. increment count of number of employees in appropriate department
3. specify relationship between employee information and department
4. initialize all appropriate fields in employee information.

*** INPUTS ***

1	hourly-employment-form	RECEIVED
2	salaried-employment-form	RECEIVED
3	tax-withholding-certificate	RECEIVED
4	hourly-employment-form	USED
5	salaried-employment-form	USED
6	tax-withholding-certificate	USED

*** OUTPUTS ***

1	hired-employee-report	GENERATED
2	hired-employee-report	DERIVED

3* process-library

These are routines used by one or more processes in the system.

193c

*** INPUTS ***

NO INPUTS FOR THIS PROCESS

*** OUTPUTS ***

FIGURE_55

PROCESS INPUT/OUTPUT

NO OUTPUTS FOR THIS PROCESS

4* salaried-employee-processing

This process produces the pay statement for salaried employees once a month.

---

1. salary defines gross pay.
2. compute taxes from gross pay.
3. subtract taxes from gross pay to obtain net pay.
4. update salaried employee information accordingly.
5. update department information accordingly.
6. generate paycheck

Note: hours worked is assumed to be 40

*** INPUTS ***

1	salaried-employee-information	USED

*** OUTPUTS ***

1	pay-statement	GENERATED
2	error-listing	GENERATED
3	salaried-employee-report	GENERATED
4	pay-statement	DERIVED
5	error-listing	DERIVED
6	salaried-employee-report	DERIVED

193d

5* terminating-emp-processing

This process deletes data, for those employees who are no longer on the payroll, from the files. It also prints a list of all employees no longer on the payroll.

FIGURE_55

PROCESS INPUT/OUTPUT

---

1. determine type of employee by employment status item
2. from this, retrieve the contents of the appropriate employee information and print in report format
3. update number of employees field in appropriate department information
4. delete employee information.

*** INPUTS ***

1	employment-termination-form	RECEIVED
2	employment-termination-form	USED

*** OUTPUTS ***

1	terminated-employee-report	GENERATED
2	terminated-employee-report	DERIVED

PUNCHED COMMENT ENTRIES

Purpose

To present selected comment entries given for one or more names in a particular data base.

Information Presented

The report presents, for those names given as input, any comment entries which are specified as parameters and are available for the names. The types of comment entries available for each name are dependent on the name type of the name the report is being generated for. The table below shows the types of comment entries that may be presented and the types of names they may be presented for.

Comment Entry Type	Name Type
DESCRIPTION	All name types
PROCEDURE	PROCESS
VOLATILITY	ENTITY
VOLATILITY-MEMBER	SET
VOLATILITY-SET	SET
DERIVATION	RELATION and SET
TRUE-WHILE	CONDITION
FALSE-WHILE	CONDITION

Format

An entry in the report is printed for each comment entry presented for each name given as input. Each entry is numbered 1*, 2*, etc. The format of each entry is:

```
**   name
        comment-entry-statement;

             [comment entry text];
```

Each line of the comment entry text is numbered within a given report entry.

Options and Alternatives

An entry in the report is produced for each comment entry (as specified by the parameters for the command generated the report) available for each name given as input. The following parameters designate the comment entries to be presented:

DERIVATION	DESCRIPTION	FALSE-WHILE
PROCEDURE	TRUE-WHILE	VOLATILITY
VOLATILITY-MEMBER	VOLATILITY-SET	

Any of these parameters prefixed with "NO" specifies that the corresponding comment entries are not to be presented in the report.

Analysis

Each name given as input to the software generating the report is searched for in the data base. If it is not found the message:

```
URA135: MAINPCOM: NAME NOT FOUND IN DATA BASE -
```

is printed. If the name is found then those comment entries (as specified by the parameters) which are available for it are printed on the report.

Usage

The report may be used by analysts to check the availability of specific types of comment entries for a class of names. For example, to check that all SETS have VOLATILITY-MEMBER, VOLATILITY-SET, and DERIVATION comment entries the commands:

```
NAME-GEN S='SET'
PUNCH-COMMENT-ENTRY VOLATILITY-MEMBER VOLATILITY-SET
DERIVATION
```

can be given.

Examples

Figure 56 presents the report for the name "salaried-employee-processing."

FIGURE_56

PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PCOM

NAME=salaried-employee-processing NODESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH

```
1*   salaried-employee-processing
     PROCEDURE;
          1              1. salary defines gross pay.
          2              2. compute taxes from gross pay.
          3              3. subtract taxes from gross pay to obtain net pay.
          4              4. update salaried employee information accordingly.
          5              5. update department information accordingly.
          6              6. generate paycheck
          7             Note: hours worked is assumed to be 40;
```

RESOURCE CONSUMPTION REPORT

Purpose

To estimate the amount of the resources that are consumed by the processors in performing a given root process.

Information Presented

The RESOURCE CONSUMPTION Report can present two kinds of information:

a) For each processor which is used to perform the given root process the following appears:

   1) The name of the processor.
   2) The frequency of processor use in terms of a user supplied interval.
   3) A list of names of the resources that the processor consumes together with the amount consumed and the unit in which the resource is measured.

b) For each resource consumed when the root process is performed the following appears:

   1) A list of names of the processors which consume the resource together with the frequency of usage in terms of the specified interval and the amount of the resource consumed.

Format

The format for the PROCESSOR and RESOURCE information is as shown in the following example:

```
PROCESSOR PETER              IS INVOKED          64 TIMES PER YEAR

  RESOURCE CONSUMED          AMOUNT CONSUMED     MEASURED IN

  R1                            145768           DOLLARS
  R2                             10512           TONS
  R3                              1644           MICROSECONDS

RESOURCE R1                  MEASURED IN DOLLARS
```

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
PETER	64	145768
SAM	52	592800
FRED	12	612
TOTALS	128	739180

Options and Alternatives

If one is only interested in a certain type of processor, the PROCESSOR-KEYWORD parameter may be specified. In this case, the processor part of the report will be broken into two sections. The first section will show those processors which have the specified keyword, and the second section will list all the other processor under the heading "OTHER PROCESSORS."

The resource part of the report will also be altered if the PROCESSOR-KEYWORD parameter is specified. For each resource, the processors which have the specified keyword are first listed and the other processors that consume the resource are listed under "OTHER PROCESSORS."

The PROCESS-LEVEL=number parameter does not change the format of the output, but it is used to control the "level of detail" of the analysis. Only as many levels as specified in this parameter will be probed in the process structure beginning at the root-process.

When the output is expected to be fairly large (i.e., contains many user-names in it), it is advantageous to specify the INDEX parameter to generate an index report that shows all the user-names that appear in the report and the report page number(s) on which they appear.

Analysis

The program searches the data base for each process name given as input. If the name is not found, the message:

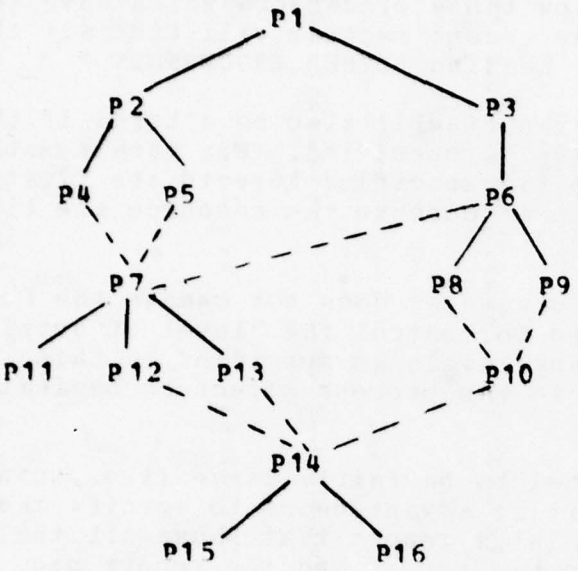
URAC59: PREPRO: ROOT PROCESS NOT FOUND IN DATA BASE -

is printed and no information is presented for that process name.

If the process is found in the data base, the program then performs a walk of the subparts-utilizes network which starts at the specified process name and extends downward to the specified number of levels. For each visit to each process in the structure, the resource consumption and processor frequency information is computed and recorded. After the network has been traversed, the recorded information is printed.

It is important to realize that a process may be visited more than once, and that its resource consumption is computed each time it is visited. This corresponds to the fact that a component process may be performed many times as a root process is performed once.

For the network shown below, where continuous lines represent a subpart relation and broken lines represent a utilizes relation, processes P1, P2, P3, P4, P5, P6, P8 and P9 are each counted once. Processes P7, P11, P12 and P13 are each counted three times. Process P10 is counted twice. And process P14, P15 and P16 are each counted eight times.



Thus, a very simple structure can generate a large number of computations.

The program expects to find the following information in the data base:

- Each process must have a frequency value. The frequency value must be convertable to "times per given INTERVAL." The program will only make one level interval conversions. For example, if the following information is in the data base:

```
YEAR  CONSISTS OF 52 WEEKS;
WEEKS CONSISTS OF 7 DAYS;
```

Then the program will convert weeks to years or years to weeks but will not convert years to days or days to years. For this, the statement:

```
YEAR CONSISTS OF 365 DAYS;
```

must be given.

- Each process must be performed by exactly one processor.

If the above conditions are not met, error messages will be printed and the analysis will be incomplete.

Usage

The report can be used to estimate the usage of the resources that are consumed by the processors in performing a given process. The estimate is accumulated for a given interval based on the user-specified resource usage parameters for each processor and the resource usage parameters which are components of the given process.

Examples

Figure 57 represents the output of the following command:

```
RCA N=hourly-employee-processing PL=50 INT=week BP BR
PK=ALL
```

Figure 58 represents the output of the following command:

```
NG s='SO=hourly-paycheck-production' NOPRINT
RCA F INT=week
```

FIGURE_57

Resource Consumption Analysis

PARAMETERS FOR: RCA

NAME=hourly-employee-processing NOINDEX PROCESS-LEVEL=50 INTERVAL=week BYPROCESSOR BYRESOURCE
PROCESSOR-KEYWORD=ALL

ROOT PROCESS: hourly-employee-processing

PROCESSOR validation-clerk IS INVOKED 150 TIMES PER week

RESOURCE CONSUMED	AMOUNT CONSUMED	MEASURED IN
paper	250	page

PROCESSOR computer-processor IS INVOKED 7000 TIMES PER week

RESOURCE CONSUMED	AMOUNT CONSUMED	MEASURED IN
cpu-time	172500	seconds
paper	862500	page

PROCESSOR payroll-processor IS INVOKED 1 TIMES PER week

RESOURCE CONSUMED	AMOUNT CONSUMED	MEASURED IN
paper	20	page
cpu-time	300	seconds

199a

FIGURE_57

Resource Consumption Analysis

RESOURCE paper MEASURED IN page

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
validation-clerk	150	250
computer-processor	7000	862500
payroll-processor	1	20
TOTALS	7151	862770

RESOURCE cpu-time MEASURED IN seconds

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
computer-processor	7000	172500
payroll-processor	1	300
TOTALS	7001	172800

1996

FIGURE_58

Resource Consumption Analysis

PARAMETERS FOR: RCA

FILE NOINDEX PROCESS-LEVEL=50 INTERVAL=week BYPROCESSOR BYRESOURCE PROCESSOR-KEYWORD=ALL

ROOT PROCESS: h-gross-pay-computation

PROCESSOR computer-processor IS INVOKED 500 TIMES PER week

RESOURCE CONSUMED	AMOUNT CONSUMED	MEASURED IN
cpu-time	15000	seconds
paper	75000	page

FIGURE_58

Resource Consumption Analysis

RESOURCE cpu-time MEASURED IN seconds

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
computer-processor	500	15000
TOTALS	500	15000

RESOURCE paper MEASURED IN page

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
computer-processor	500	75000
TOTALS	500	75000

1994

FIGURE_58

Resource Consumption Analysis

ROOT PROCESS: total-hours-computation

PROCESSOR computer-processor IS INVOKED 500 TIMES PER week

RESOURCE CONSUMED	AMOUNT CONSUMED	MEASURED IN
cpu-time	15000	seconds
paper	75000	page

1996

FIGURE_58

Resource Consumption Analysis

RESOURCE cpu-time MEASURED IN seconds

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
computer-processor	500	15000
TOTALS	500	15000

RESOURCE paper MEASURED IN page

PROCESSOR CONSUMING	FREQUENCY	AMOUNT CONSUMED
computer-processor	500	75000
TOTALS	500	75000

J66I

SECURITY ANALYSIS REPORT

Purpose

To idenfity security conflicts in the design of the target system.

Information Presented

For each name received as input, the report prints a message for each security conflict associated with that name and then, at the end of the report, prints a matrix which summarizes the security conflicts encountered for all the names received as input.

Optionally, the report also produces a list of those names in the data base for which security information is possible but for which such information does not exist.

Format

The format of the report is self-explanatory.

Options and Alternatives

There are only two options:

1) Whether or not to have an index produced.
2) Whether or not to include a list of names without security information.

Analysis

The program searches the data base for each name given as input. If the name is not found, the message:

URA195: MAINSECA : NAME NOT FOUND IN DB - user-name

is printed an no information is presented for that name. The type of the name is then checked. If its type does not make sense in the context of the Security Analysis Report the message:

URA203: MAINSECA : INPUT NAME HAS INVALID TYPE

is printed and no information is presented for that name.

If the name is a data type (i.e., SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT) the following steps are taken:

1) For every name contained in the given name, every classification of the contained name is checked to see that the given (containing) name has the same classification at a level greater than or equal to the level of the contained name.

2) For every PROCESS, PROCESSOR or INTERFACE which accesses the given name, every classification of the given name is checked to see that the accessor has the same classification at a level greater than or equal to the level of the given (accessed) name.

If a classification is found missing, or a level mismatch occurs in one of the above steps, an appropriate message is printed which explains the security conflict.

If the given name is a PROCESS, PROCESSOR or INTERFACE, then the following occurs for each name in the subparts structure of the given name.

Each classification of each data item that the name accesses is checked to see that the accessing name has the same classification at a level greater than or equal to the level of the accessed item. If not, an appropriate message is printed.

Usage

The report can be used to check the consistency of the security provisions of the target system and to identify components of the system for which no security provisions exist.

Examples

Figure 59 represents the output of the following command:

```
SECA N=hourly-paycheck-validation PNSI PMAT NOINDEX
```

FIGURE_59

Security Analysis Report

PARAMETERS FOR: SECA

NAME=hourly-paycheck-validation PRINT-NO-SECURITY-INFORMATION PRINT-MATRIX NOINDEX

The GROUP address has no security information.

The ELEMENT age has no security information.

The ELEMENT apartment-number has no security information.

The ELEMENT birthdate has no security information.

The GROUP check has no security information.

The ELEMENT check-number has no security information.

The ELEMENT city has no security information.

The PROCESSOR computer-processor has no security information.

The ELEMENT count-of-hourly-employees has no security information.

The ELEMENT count-of-salaried-employees has no security information.

The ELEMENT cumulative-federal-deductions has no security information.

The ELEMENT cumulative-fica-deductions has no security information.

The ELEMENT cumulative-gross-pay has no security information.

The ELEMENT cumulative-hours has no security information.

The ELEMENT cumulative-state-deductions has no security information.

The ELEMENT cumulative-tax-deductions has no security information.

The ELEMENT current-date has no security information.

The ELEMENT department has no security information.

FIGURE_59

Security Analysis Report

The SET department-file has no security information.

The PROCESS department-file-addition has no security information.

The PROCESS department-file-removal has no security information.

The ENTITY department-information has no security information.

The GROUP department-update-data has no security information.

The INTERFACE departments-and-employees has no security information.

The GROUP emp-termination-data has no security information.

The INPUT employee-information has no security information.

The GROUP employee-name has no security information.

The ELEMENT employment-date has no security information.

The ELEMENT employment-status has no security information.

The ELEMENT error-code has no security information.

The OUTPUT error-listing has no security information.

The PROCESS federal-deductions-update has no security information.

The ELEMENT federal-tax has no security information.

The PROCESS fica-deductions-update has no security information.

The ELEMENT fica-tax has no security information.

The ELEMENT first-name has no security information.

The PROCESS funds-update has no security information.

The ELEMENT gross-pay has no security information.

The PROCESS gross-pay-update has no security information.

FIGURE_59

Security Analysis Report

The GROUP h-derived-pay-data has no security information.

The GROUP h-emp-report-entry has no security information.

The PROCESS h-gross-pay-computation has no security information.

The PROCESS h-report-entry-generation has no security information.

The PROCESS hire-report-entry-generation has no security information.

The OUTPUT hired-employee-report has no security information.

The GROUP hired-report-entry has no security information.

The PROCESS hourly-emp-processing has no security information.

The PROCESS hourly-emp-update has no security information.

The PROCESS hourly-employee-processing has no security information.

The OUTPUT hourly-employee-report has no security information.

The INPUT hourly-employment-form has no security information.

The PROCESS hourly-information-creation has no security information.

The PROCESS hourly-information-deletion has no security information.

The GROUP hourly-job-data has no security information.

The PROCESS hourly-paycheck-production has no security information.

The ELEMENT hours-per-day has no security information.

The PROCESS hours-update has no security information.

The ELEMENT house-number has no security information.

The ELEMENT initial has no security information.

FIGURE_59

Security Analysis Report

The ELEMENT job-number has no security information.

The ELEMENT job-title has no security information.

The ELEMENT net-pay has no security information.

The PROCESS net-pay-computation has no security information.

The PROCESS new-employee-processing has no security information.

The ELEMENT number-of-deductions has no security information.

The ELEMENT number-of-employees has no security information.

The ELEMENT overtime-hours-worked has no security information.

The PROCESS pay-computation-validation has no security information.

The ELEMENT pay-date has no security information.

The ELEMENT pay-rate has no security information.

The GROUP pay-stub has no security information.

The INTERFACE payroll-department has no security information.

The SET payroll-master-information has no security information.

The PROCESS payroll-processing has no security information.

The PROCESSOR payroll-processor has no security information.

The OUTPUT paysystem-outputs has no security information.

The GROUP personal-data has no security information.

The ELEMENT phone has no security information.

The PROCESS process-library has no security information.

The ELEMENT regular-hours-worked has no security information.

FIGURE_59

Security Analysis Report

The ELEMENT remaining-funds has no security information.

The GROUP s-derived-pay-data has no security information.

The GROUP s-emp-report-entry has no security information.

The PROCESS s-gross-pay-computation has no security information.

The PROCESS s-report-entry-generation has no security information.

The GROUP salaried-emp-pay-data has no security information.

The PROCESS salaried-emp-update has no security information.

The SET salaried-employee-file has no security information.

The ENTITY salaried-employee-information has no security information.

The PROCESS salaried-employee-processing has no security information.

The OUTPUT salaried-employee-report has no security information.

The INPUT salaried-employment-form has no security information.

The PROCESS salaried-information-creation has no security information.

The PROCESS salaried-information-deletion has no security information.

The GROUP salaried-job-data has no security information.

The PROCESS salaried-paycheck-production has no security information.

The PROCESS salaried-paycheck-validation has no security information.

The ELEMENT salary has no security information.

The ELEMENT sex has no security information.

The ELEMENT state has no security information.

FIGURE_59

Security Analysis Report

The PROCESS state-deductions-update has no security information.

The ELEMENT state-tax has no security information.

The ELEMENT street has no security information.

The ELEMENT supervisor has no security information.

The ELEMENT surname has no security information.

The PROCESS tax-computation has no security information.

The INPUT tax-withholding-certificate has no security information.

The GROUP term-report-entry has no security information.

The PROCESS term-report-entry-generation has no security information.

The ELEMENT termination-date has no security information.

The ELEMENT total-budget has no security information.

The ELEMENT total-deductions has no security information.

The PROCESS total-deductions-computation has no security information.

The ELEMENT total-hours has no security information.

The PROCESS total-hours-computation has no security information.

The ELEMENT zip-code has no security information.

The PROCESS hourly-paycheck-validation accesses the GROUP time-card-data
but does not have the security access right classified. It must have this security access right at a l
0 or higher in order to access time-card-data.

201f

The PROCESS hourly-paycheck-validation accesses the GROUP time-card-data and has the security access ri
top-secret at a level of 2. However, it must have this security access right
at a level greater than or equal to 5 in order to access time-card-data.

FIGURE_59

Security Analysis Report

The PROCESS hourly-paycheck-validation accesses the GROUP hourly-emp-pay-data
but does not have the security access right classified. It must have this security access right at a lev
0 or higher in order to access hourly-emp-pay-data.

The PROCESS hourly-paycheck-validation accesses the GROUP hourly-emp-pay-data and has the security access
top-secret at a level of 2. However, it must have this security access right
at a level greater than or equal to 5 in order to access hourly-emp-pay-data.

The PROCESS hourly-paycheck-validation accesses the GROUP error-listing-entry
but does not have the security access right classified. It must have this security access right at a lev
0 or higher in order to access error-listing-entry.

The PROCESS time-card-validation accesses the ELEMENT employee-identification-number
but does not have the security access right classified. It must have this security access right at a le
0 or higher in order to access employee-identification-number.

The PROCESS time-card-validation accesses the ELEMENT social-security-number
but does not have the security access right classified. It must have this security access right at a le
0 or higher in order to access social-security-number.

The PROCESS time-card-validation accesses the ELEMENT status-code but does not have the security access
classified. It must have this security access right at a level of 0 or higher in order to access status

FIGURE_59

Security Analysis Report

In the following matrix, the row names represent data items with which the column names are in conflict.

An X in entry (i,j) signifies that the name whose column number is j does not have a high enough security level (for one or more classifications) to access or contain the name whose row number is i.

An M in entry (i,j) signifies that the name whose column number is j is missing at least one access right or classification that it must have in order to access or contain the name whose row number is i.

A B in entry (i,j) signifies that both the above conditions hold.

If a blank occurs in entry (i,j), the names involved have no security conflicts.

ROW NAMES		COLUMN NAMES	
1 time-card-data	GROUP	1 hourly-paycheck-validation	PROCESS
2 hourly-emp-pay-data	GROUP	2 time-card-validation	PROCESS
3 error-listing-entry	GROUP		
4 employee-identification-number	ELEMENT		
5 social-security-number	ELEMENT		
6 status-code	ELEMENT		

FIGURE_59

Security Analysis Report



STRUCTURE

Purpose

To present the implied hierarchy of PROCESSES, INPUTS, OUTPUTS, INTERFACES or PROCESSORS defined in the Analyzer data base, from the use of the SUBPARTS statements relating them.

Information Presented

The report presents all SUBPARTS structures for a given class of names (INTERFACES, INPUTS, OUTPUTS, PROCESSES or PROCESSORS) as specified by the parameters when generating the report.

The structures start with all names which are not PART of any higher structure. These names are designated level 1 names. The SUBPARTS of level 1 names are presented as level 2 names. The SUBPARTS of the level 2 names are then presented and so on.

Format

The report presents the structures under three headings: COUNT, LEVEL, and NAME. NAME presents the name of the object in the structure, LEVEL presents the level number associated to the name corresponding to its position in the structure and COUNT presents the position (line) in the report where the name is printed out. Each level is indented (as specified by the INDENT parameter) to further accent the idea of structure.

A summary section for the report provides a count (under the COUNT heading) of the number of names presented at a given level (as designated by the LEVEL heading).

Options and Alternatives

The INPUT, OUTPUT, PROCESS, INTERFACE and PROCESSOR parameters for the command specify which type of names the report will be produced for. One and only one of these parameters may be specified for the command producing the report.

The number of spaces which each level of the structure is indented may be assigned by the INDENT parameter. If no value is given INDENT defaults to 3, but may take on any value from 1 to 10.

An INDEX for the report is produced when the INDEX parameter is used.

Analysis

A check is made that at least one name (of the name type designated by the parameters) exists in the data base which is not PART of a larger structure. If no such name exists the message:

URA288: STATPS: NO NAMES AT LEVEL ONE -

is printed. For each name found, its SUBPARTS structure is traced and presented in the report.

If a loop is encountered in the structure the message:

URA285: ERRPS: THE FOLLOWING NAMES ARE INVOLVED IN LOOPS -

is printed with the names involved.

Should the structure consist of more than 50 levels (the limit that the software has been designed to handle) the message:

URA284: MAINSTR: TOO MANY LEVELS - CONTINUING -

is printed.

Usage

The report is an aid to analysts in maintaining consistency in structures defined for the target system description. Especially where a top-down approach is being used, the analyst is concerned that names have been inserted into the proper level of a strucutre.

Examples

Figure 60 presents a STRUCTURE report for INPUT names. Figures 61, 62, 63 and 64 present STRUCTURE reports for OUTPUT, INTERFACE, PROCESS, and PROCESSOR names, respectively.

FIGURE_60

INPUT STRUCTURE

```
PARAMETERS FOR: STR

   INPUT INDENT=3 NOINDEX

COUNT LEVEL NAME

    1     1 employee-information
    2       2   time-card
    3       2   hourly-employment-form
    4       2   salaried-employment-form
    5       2   tax-withholding-certificate
    6       2   employment-termination-form



LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT
    1     1       2     5
```

FIGURE_61

OUTPUT STRUCTURE

```
PARAMETERS FOR: STR

   OUTPUT INDENT=3 NOINDEX

COUNT LEVEL NAME

    1     1 paysystem-outputs
    2      2   pay-statement
    3      2   error-listing
    4      2   hourly-employee-report
    5      2   salaried-employee-report
    6      2   hired-employee-report
    7      2   terminated-employee-report



LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT
    1     1       2     6
```

FIGURE_62

INTERFACE STRUCTURE

```
PARAMETERS FOR: STR

   INTERFACE INDENT=3 NOINDEX

COUNT LEVEL NAME

    1     1 departments-and-employees
    2      2   employee
    3      2   payroll-department



LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT
    1     1       2     2
```

FIGURE_63

PROCESS STRUCTURE

PARAMETERS FOR: STR

PROCESS INDENT=3 NOINDEX

```
COUNT LEVEL NAME

    1     1 hourly-emp-processing

    2     1 payroll-processing
    3      2   new-employee-processing
    4        3     salaried-information-creation
    5        3     hourly-information-creation
    6        3     hire-report-entry-generation
    7        3     department-file-addition
    8      2   terminating-emp-processing
    9        3     salaried-information-deletion
   10        3     hourly-information-deletion
   11        3     term-report-entry-generation
   12        3     department-file-removal
   13      2   hourly-employee-processing
   14        3     hourly-paycheck-validation
   15          4       time-card-validation
   16        3     hourly-emp-update
   17          4       hours-update
   18        3     h-report-entry-generation
   19        3     hourly-paycheck-production
   20          4       h-gross-pay-computation
   21          4       total-hours-computation
   22      2   salaried-employee-processing
   23        3     salaried-paycheck-validation
   24        3     salaried-emp-update
   25        3     s-report-entry-generation
   26        3     salaried-paycheck-production
   27          4       s-gross-pay-computation
   28      2   process-library
   29        3     pay-computation-validation
   30        3     tax-computation
   31        3     net-pay-computation
   32        3     total-deductions-computation
   33        3     gross-pay-update
```

FIGURE_63

PROCESS STRUCTURE

COUNT	LEVEL	NAME
34	3	federal-deductions-update
35	3	state-deductions-update
36	3	fica-deductions-update
37	3	funds-update

LEVEL	COUNT	LEVEL	COUNT	LEVEL	COUNT	LEVEL	COUNT	LEVEL	COUNT
1	2	2	5	3	25	4	5		

FIGURE_64

PROCESSOR STRUCTURE

```
 PARAMETERS FOR: STR

    PROCESSOR INDENT=3 NOINDEX

COUNT LEVEL NAME


    1     1 payroll-processor
    2      2   validation-clerk
    3      2   computer-processor



LEVEL COUNT    LEVEL COUNT    LEVEL COUNT    LEVEL COUNT    LEVEL COUNT
    1     1        2     2
```

SUMMARY OF THE DATA BASE

Purpose

This report provides statistical information with respect to the usage of different name types (e.g., PROCESSES, ELEMENTS, etc.) and can be used as an aid in estimating size of the Language description in an Analyzer data base.

Information Presented

The report provides an entry for each different name type (PROCESS, ELEMENT, etc.) defined in a particular data base. For example, if only PROCESSES have been defined, only a PROCESS entry will appear in the report. For each name type the report entry presents:

- The number of names in the data base of that name type (COUNT)
- The number of these names that have SYNONYMS (#W/SYN)
- The percentage of these names with SYNONYMS (PERCENT)
- The number of these names that have a DESCRIPTION (#W/DESC)
- The percentage of these names with a DESCRIPTION (PERCENT)

An entry is also provided consisting of all of the above considering all name types (**TOTAL**) in the data base.

Format

The report presents six columns of information as follows:

```
name-type COUNT #W/SYN PERCENT #W/DESC PERCENT
```

in a table format. A row of the table consists of values for each of the above columns. A row is presented for each name type defined in the data base and the name types are presented in alphabetical order. A row is also printed presenting the total for each column. Since there are a limited number of name types, the report always fits on one page.

Options and Alternatives

There are no options available for this report.

Analysis

The software generating the report looks at every name in the data base and updates three fields according to its name type (COUNT), whether or not it has a DESCRIPTION (#W/DESC), and whether or not it has any SYNONYMS (#W/SYN). After all names have been looked at, percentages (PERCENT) for these names with DESCRIPTIONS and SYNONYMS are computed within each name type.

Finally, totals are produced for values retrieved for each table heading and are printed.

If the report is generated for an empty data base, the message:

NO NAMES IN D.B.-

is printed.

Usage

The major benefit of this report is to project management personnel in recording progress being made in the target system description procedure. Comparing a particular DATA BASE SUMMARY with previous summaries allows management to see where some of the work effort has been concentrated. For example, if 52 PROCESSES are counted in the report compared to 12 previously, it shows that the major project effort has been centered around definition of PROCESSES.

General progress may be evaluated by looking at the totals presented for successive reports.

Examples

Figure 65 presents the SUMMARY generated from a description of a target system.

FIGURE_65

DATA BASE SUMMARY

	COUNT	#W/SYN	PERCENT	#W/DESC	PERCENT
*** UNDEFINED ***	2	0		0	
ATTRIBUTE	9	0		4	44.44
ATTRIBUTE-VALUE	11	0		1	9.09
CONDITION	6	0		0	
ELEMENT	52	0		12	23.08
ENTITY	3	3	100.00	3	100.00
EVENT	13	0		4	30.77
GROUP	19	0		5	26.32
INPUT	6	6	100.00	6	100.00
INTERVAL	4	0		0	
KEYWORD	5	0		0	
MAILBOX	1	0		0	
MEMO	7	0		3	42.86
OUTPUT	7	7	100.00	7	100.00
PROBLEM-DEFINER	13	0		0	
PROCESS	37	1	2.70	8	21.62
INTERFACE	3	3	100.00	3	100.00
RELATION	2	0		2	100.00
SECURITY	3	0		0	
SET	4	4	100.00	4	100.00
SOURCE	5	0		0	
SYSTEM-PARAMETER	12	0		5	41.67
UNIT	2	1	50.00	1	50.00
PROCESSOR	3	1	33.33	1	33.33
RESOURCE	2	1	50.00	1	50.00
RESOURCE-USAGE-PARAMETER	3	1	33.33	1	33.33
CLASSIFICATION	4	0		0	
** TOTAL **	238	28	11.76	71	29.83

INDEX

Purpose

To provide a reference into a particular URA report to locate all occurrences of the use of a particular user defined name in the report.

Information Presented

The report presents all the user defined names used in a given URA report, which must be one of the:

CONTENTS REPORT
DICTIONARY REPORT
EXTENDED PICTURE
FORMATTED-PROBLEM-STATEMENT
PICTURE
PROCESS CHAIN
PROCESS INPUT/OUTPUT or
STRUCTURE,

the page numbers in the report where each name occurs, and the number of times the name appears within the pages it occurs (if more than once).

Format

The report consists of an entry for each name used in one of the above reports. The entry consists of:

-the name presented in the report.

-the page numbers (separated by commas) that the name occurs on in the report.

-the number of occurrences (enclosed in brackets) of the name on a particular page.

Each entry is numbered and the entries are arranged in alphabetical order by name.

Options and Alternatives

There are no options for this report.

Analysis

The input for the software producing the INDEX is obtained from

one of the report-producing modules which allows the INDEX parameter. If no input is available, because the report presents no information, the message:

URA287: MAINIDX: NO NAMES IN INDEX

is printed if input is available (in the form of names and line numbers) the names are sorted and presented as the INDEX report.

Usage

The INDEX is intended as a reference into a report for purposes of locating all occurrences of the use of a particular name in the report. The INDEX is usually desirable whenever a report over a few pages in size is to be generated.

Examples

Figure 66 presents a PROCESS INPUT/OUTPUT REPORT with an INDEX.

FIGURE_66

PROCESS INPUT/OUTPUT

PARAMETERS FOR: PRIO

NAME=payroll-processing INPUT OUTPUT DESCRIPTION NOPROCEDURE NONEW-PAGE INDEX PRINT NOPINCH

1* payroll-processing

This process represents the highest level process in the target system. it accepts and processes all inputs and produces all outputs.

*** INPUTS ***

1	employee-information	RECEIVED
2	employee-information	USED
3	payroll-master-information	USED

*** OUTPUTS ***

1	paysystem-outputs	GENERATED
2	paysystem-outputs	DERIVED
3	payroll-master-information	UPDATED

FIGURE_66

INDEX

1 employee-information 132(2)

2 payroll-master-information 132(2)

3 payroll-processing 132

4 paysystem-outputs 132

PART IV

USER REQUIREMENTS ANALYZER

COMMAND DESCRITIONS

A Note on Version 3.2 of URA

There are several important changes that have been made in URA which makes Version 3.2 different than Version 2.1.

First, several internal modifications (which are transparent to the user) have been made to make the URA more efficient. In particular, a data base search index has been added. This index provides a more efficient method for the retrieval of names contained in the data base.

Several changes have been made to the NAME GEN command. These changes are:

1) the replacement of the name grouping parameters (e.g., PROCESS, KEY=user-name) by the SELECTION parameter.

2) the addition of the ATTRIBUTE-VALUE ordering option to the ORDER parameter.

The DISPLAY control command has been added.

The MTS command has been implemented to allow the execution of Multics commands from URA mode. The new report commands RESOURCE-CONSUMPTION-ANALYSIS and SECURITY-CONSISTENCY-ANALYSIS have been added. The reports for Structure, Contents, Summary, and Formatted-Problem-Statement have been changed to accommodate the new object types in the data base.

Names for objects may now consist of certain special characters (see Appendix I).

The HELP command information has been updated to reflect these changes.

THE URA COMMAND LANGUAGE

The URA Command Language consists of three basic types of commands:

Report Commands

Modifier Commands

Control Commands

Report Commands retrieve data from the URA data base and output it in some meaningful format. These reports do not change the contents of the data base whatsoever. Their purpose is solely that of displaying orderings and/or relationships within the current problem statement. The following Report Commands are available in this version of the Analyzer:

CONSISTS-COMPARISON
CONSISTS-MATRIX
CONTENTS
DATA-PROCESS
DICTIONARY
ENTITY-IDENTIFIER
EXTENDED-PICTURE
FORMATTED-PROBLEM-STATEMENT
FREQUENCY
KWIC
NAME-GEN
NAME-LIST
PICTURE
PRINT-ATTRIBUTE-VALUES
PROCESS-CHAIN
PROCESS-INPUT-OUTPUT
PUNCH-COMMENT-ENTRY
RESOURCE-CONSUMPTION-ANALYSIS
SECURITY-CONSISTENCY-ANALYSIS
STRUCTURE
SUMMARY

Modifier Commands are intended to modify the contents of the URA data base in the manner defined by the problem definer. These commands take legal URL statements or URL names as input. URA then generates error diagnostics as well as an output report to present the outcome of the data base alteration. The following Modifier Commands are available in this version of the Analyzer:

CHANGE-TYPE
DELETE
DELETE-COMMENT-ENTRY
DELETE-PSL
INPUT-PSL

RENAME
REPLACE-COMMENT-ENTRY

Control Commands are the means to pass certain control information to the User Requirements Analyzer. The SET command, for example, allows the user to define which URA data base is to be accessed by the Report and Modifier Commands as well as setting various switches and assigning input and output files. Control Commands are installation dependent and therefore are given in Appendix F.

Although any of the commands can be issued independently of each other, it is often advantageous to use some commands in sequence. This means that output of one command can be used as input by another. The most common instance of this is when NAME-GEN is used to select certain names (say all PROCESSES for example) which can then be used as input to a Report Command (possibly PICTURE, for a PICTURE REPORT for all PROCESS names).

URA Command Language/Installation Dependencies

URA is a software system that is designed to be used interactively in a time sharing system environment. It contains its own data base management system but it is dependent on the time sharing operating system for the usual facilities of sign on, identification and security, file creation and editing, etc.

Differences in operating systems and operations at particular installations affect the way in which URA is executed at a particular installation. There are basically three aspects of URA that are installation "dependent":

1) Control commands

2) Method of initiating and executing URA

3) File names used by URA

These dependencies are presented in the addendum for the particular installation.

URA can also be used in batch mode at most installations. The commands necessary to accomplish this are also installation dependent and are not covered in this paper.

Command Language Syntax Notation

ABBREVIATIONS

To enable the user to fit a lengthy command on the alotted line and eliminate some of the tedium of command specification,

abbreviated forms for both commands and parameters may be used. Each abbreviation can be found in parentheses immediately following the word it represents. For example, the command:

CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT can be written as

CT N=GROSS-PAY T=ELEMENT

BLANKS

A blank must appear between the command and any accompanying parameter, and between successive parameters. Several blanks are treated as a single blank and may be inserted whenever a single blank is necessary. For example:

CT N=GROSS-PAY T=ELEMENT can be written as

CT N=GROSS-PAY T=ELEMENT

BRACES

In the following examples, when parameters or parameter values are enclosed in braces ({ }), a choice among the two or more entries must be made. It is important to note that one and only one of the options must be chosen. For example, the braces used in describing the syntax of the CHANGE-TYPE command specifies that the command must either be of the form:

CT N=user-name T=name-type

or

CT F=fdname [T=name-type]

BRACKETS

Whenever parameter notation in an example appears within brackets ([ ]), it indicates a feature the user may optionally use. For example, the TYPE parameter in the CHANGE-TYPE command is optional when the FILE parameter is also used. Therefore, the command may be of the form:

CT FILE=fdname TYPE=name-type

or

CT FILE=fdname

The syntax of the FILE parameter shows that the parameter may be given either as:

FILE=fdname

or just

FILE

No other variations are acceptable (except those already specified, i.e., abbreviations, etc.).

COMMAND LINE

Each command must appear on a separate line and totally on that line. A command cannot be split on succeeding lines. Only columns 1 through 72 of each line can be used.

GENERAL COMMAND SYNTAX

The command identifiers (name of the command) must precede any accompanying parameter or list of parameters.

COMMAND PARAMETERS

Parameters for a command separated by one or more blanks. Parameters may be given in any order, but are processed from left to right. If conflicting parameters are used, the right-most parameter, i.e., the last one given, is considered to be correct and is the one used in the processing of the command.

ELLIPSIS

The ellipsis (...) signifies that the command construct immediately preceding the ellipsis can be repeated as many times as desired by the user.

INTEGERS

The integers required for parameters must be positive integers. If a value range is given for a particular parameter description, that restriction must also be met.

NAMES

All user defined names (user-names) must be formed from the legal character set presented in the Appendix B. Note the following:

- A name can be any combination of thirty or less code 3 or code 4 characters where the first character is a letter or

other code 3 character.

- Blanks cannot be used in the name.

- A user defined name cannot be a URL RESERVED WORDS. For a list of Reserved words see Appendix B of Part II, URL User's Maual.

For example,

GROSS-PAY, EMPLOYEE-NUMBER, #-of-UNITS @ $.06/lbs.

are all legal names.

PROCESS, 123-HILL-STREET, WHY?, and BL ANK

are illegal names. "PROCESS" cannot be used as a user-name because it is a URL Reserved Word. "123-HILL-STREET" is illegal because it starts with an integer rather than a letter. "WHY" is illegal because it contains the "?" character which is not allowed, and "BL ANK" contains an imbedded blank.

CAPITALIZATION

In this working paper all URA commands, parameters and reserved words appear in upper case. In some systems, such as Multics, where the standard command convention is lower case, the user should use lower case for all URA commands, parameters, and reserved words.

Format of Command Descriptions

All the URA commands in this paper are described in the following format:

Command: COMMAND-NAME Type: command type

Purpose: This presents the function of the command in the URA system whether it generates a report, modifies the data base or gives control information to URA. (The "URA Users Manual" and "URA Reports" present detailed descriptions of the reports generated by each command.)

Prototype: This presents the legal syntax for the command. The Command Language Syntax Notation specifies what the special symbols (such as braces and brackets) represent in interpreting the syntax.

Parameters:
For each parameter available for the command, this section provides a brief description explaining how

the parameter changes the action of the command. (Part I and Part III explain how to use these parameters effectively.)

There are basically five types of parameters:

Input data parameters - these parameters specify the data to be used as input to the command. FILE and NAME are examples of Input data parameters.

Input control parameters - these parameters specify how the input data is to be used, changed, etc., by the command. The TYPE parameter for the CHANGE-TYPE command and CONTAINED/CONSISTS parameter for the CONSISTS-MATRIX command are examples of this type.

Output data parameters - these parameters specify if output is to be generated from the command and the form in which it is presented. The PUNCH an PRINT parameters are examples of this type of parameter.

Output option parameters - these parameters specify options which may be included or omitted from the output. The LEVELS parameter for the CONTENTS command and the DESCRIPTION parameter for the DICTIONARY command are examples of this type.

Output format parameters - these parameters specify alternate formats for the presenting the information in the output from the command. The NEW-PAGE parameter and HMARG parameter for the FPS command are examples of this.

The parameters for each command will be presented in the above order according to type.

Defaults: These present which parameters will be used for the command, or what value a parameter will have, if the parameter, or value, is not explicitly defined. For example, if

CONTENTS

is specified, the defaults for this command are such that this has the same effect as specifying:

CONTENTS FILE NOINDEX LEVELS=ALL NONCFLAG

If a "no default" is given, this means that if not explicitly defined, the corresponding parameter will

not be used for the command.

Messages: These are the possible error messages that may occur if the parameters for the command are not specified correctly. The "URA Users Manual" explains what to do should these messages be encountered.

Examples: Actual example of the command syntax are presented. (The results from these examples are presented in Part I and Part III.)

Command: CHANGE-TYPE Type: modifier command

Purpose: To change the name type of a user name defined in the user's data base. A record of this change is generated in the form of the CHANGE-TYPE-REPORT.

Prototype: CHANGE-TYPE(CT) {NAME(N)=user-name TYPE(T)=name-type}
{FILE(F)[=fdname] [TYPE(T)=name-type]}

Parameters:

Input-data FILE(F)[=fdname] Default: no default

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The file format for each line of the input file must be of the form:

user-name [name-type]

Free format is allowed so the user-name does not have to start in the first position in the line. The two names must be separated by one or more blanks. The name-type is optional. If a name-type is not specified for each user-name, the name type for each of these names will be changed to the type specified in the TYPE parameter. One of these alternative methods of assigning a type must be used, but not both. If both are used, all the names in the file will be assigned the name type specified by the TYPE parameter.

NAME(N)=user-name Default: no default

The given user-name is the name for which the change is to be made. When the NAME parameter is used, the TYPE parameter must be used in conjunction with it.

Input-Control TYPE(T)=name-type Default: no default

This parameter specifies the new name type to be used in the change. See Appendix E of "The User Requirements Language, Language Reference Manual"[2] for a list of all possible name types.

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.
[2] Part II, URL User's Manual.

Messages: If neither FILE nor NAME parameter are given, the error message:

NO NAME GIVEN

will be generated by URA. If one of these two parameters are given, but no TYPE is specified, the error message:

NO TYPE GIVEN WITH "NAME=" OR "FILE" PARAMETER

will be given.

Examples: CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT

CT F T=ELEMENT

CT FILE T=GROUP

Command: CONSISTS-COMPARISON Type: report command

Purpose: To produce the CONSISTS-COMPARISON REPORT.

Prototype: CONSISTS-COMPARISON(CNC) [parameter]...

Parameters:

Input-Data FILE(F)[=FDNAME] Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. In any case, the names in the input file must be SET, INPUT, OUPUT, ENTITY and/or GROUP names. The format of the input file must be one name per line.

Examples: CNC

CNC FILE

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: CONSISTS-MATRIX Type: report command

Purpose: To produce the CONSISTS MATRIX REPORT.

Prototype: CONSISTS-MATRIX(CM) {CONTAINED(CNTD)}
{CONSISTS(CSTS) } [parameter]...

Parameters:

Input-Data FILE(F)[=FDNAME], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is specified via the NAME parameter, the report is produced only for that name. The format of the input file must be one name per line.

Input-Control CONTAINED(CNTD), CONSISTS(CSTS) Default: no default

Since no default exists, one of the above must be specified. If CONTAINED is given, the names used as input must be ELEMENT, GROUP, ENTITY, INPUT and/or OUTPUT names. If the CONSISTS parameter is given the names used as input must be SET, ENTITY, INPUT, OUTPUT and/or GROUP names.

Messages: If neither CONTAINED nor CONSISTS is specified, the message:

MUST GIVE EITHER CONSISTS OR CONTAINED PARAMETER

will be printed.

Examples: CM N=EMPLOYEE-NUMBER CNTD

CM FILE CSTS

CM CSTS

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: CONTENTS Type: Report command

Purpose: To produce the CONTENTS REPORT.

Prototype: CONTENTS(CONT) [parameter]...

Parameters:

Input-Data FILE(F)[=fdname],NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced for all the names in the file. When a name is specified by the NAME parameter, the report is produced for that name alone. In any case, the names used as input to the command must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. The format of the input file must be one name per line.

Output-Data INDEX,NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the report consisting of an alphabetical listing of all names used in the report and the pages on which they occur.

Output-Option LEVELS={integer} Default: LEVELS=ALL
{ALL }

The LEVELS parameter specifies the lowest level of subordinate names to be outputted. The ALL value indicates that all subordinate names should be outputted. LEVELS can take on any integer value from 1 to 50.

NOFLAG,NONCFLAG Default: NONCFLAG

The NOFLAG parameter flags all GROUPS in the output reports that do not CONSIST of anything else, and those undefined names which are contained in a GROUP, INPUT, OUTPUT, ENTITY or SET.

PRINT-SECURITY-INFORMATION(PSI),
NO-PRINT-SECURITY-INFORMATION(NPSI)
Default: PSI

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

The P$I parameter requests that the classification of each entry be printed.

Examples: CONTENTS N=VARYING-EMPLOYEE-DATA

CONT P

Command: DATA-PROCESS Type: report command

Purpose: To produce the DATA PROCESS REPORT.

Prototype: DATA-PROCESS(DP)
{DATA(D) } [parameter]...
{PROCESS(P)}

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The format of the input file must be one name per line.

When a name is given via the NAME parameter, the report is produced only for that name.

Input-Control DATA(D), PROCESS(P) Default: no default

Since no default exists, one of the above must be specified. If DATA is specified, the names used as input to the comand must be SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. If PROCESS is specified, the names used as input to the command must be PROCESS names.

Output-Options DPMAT, NODPMAT Default: DPMAT

With the DPMAT parameter in effect, the DATA PROCESS INTERACTION MATRIX is presented as part of the report. When NODPMAT is specified, this matrix is not printed.

DPANL, NODPANL Default: DPANL

With the DPANL parameter in effect, analysis is done on the DATA PROCESS INTERACTION MATRIX (whether printed or not) and presented as the DATA PROCESS INTERACTION ANALYSIS. When NODPANL is specified, this analysis is not done.

PMAT, NOPMAT Default: PMAT

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

With the PMAT parameter in effect, the PROCESS INTERACTION MATRIX is presented as part of the report. When NOPMAT is specified, this matrix is not printed.

PANL, NOPANL Default: PANL

With the PANL parameter in effect, analysis is done on the PROCESS INTERACTION MATRIX (whether printed or not) and presented as the PROCESS INTERACTION MATRIX ANALYSIS. When NOPANL is specified, this analysis is not done.

Messages: If neither DATA nor PROCESS is specified, an error message:

MUST GIVE EITHER "DATA" OR "PROCESS" PARAMETER

will be printed.

Examples: DP N=PAYROLL-PROCESSING PROCESS

DP DATA

Command: DELETE Type: modifier command

Purpose: To delete a name or list of names from the data base. When a name is deleted all of its connections to other names in the data base are also deleted. A permanent record of the change is also generated in the form of the DELETION REPORT.

Prototype: DELETE(DEL)
{FILE(F)[=fdname]}
{NAME(N)=user-name}

Parameters:

Input Data FILE(F)[=fdname], NAME(N)=user-name
Default: no default

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and all names in the file are deleted from the data base. The format of the input must be one name per line. When a name is specified by the NAME parameter, that name is deleted from the data base.

Messages: If neither the FILE nor NAME parameter is specified, the message:

NO NAME OR FILE WAS SPECIFIED

will be given.

Examples: DELETE N=FIELD-CHECK-NEW

DEL FILE

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: DELETE-COMMENT-ENTRY Type: modifier command

Purpose: To delete from the data base, for the given name or list of names, those comment entries associated with each comment entry statement specified in the list of parameters. A permanent record of the change is generated in the form of the DELETED COMMENT ENTRIES report.

Prototype: DELETE-COMMENT-ENTRY(DCOM)
{FILE(F)[=fdname]}
{NAME(N)=user-name}[parameter]...

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name
Default: no default

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, only the specified comment entries for that name are deleted. Either FILE or NAME must be given but not both. The format of the input file must be one name per line.

Input-Control When given as parameters, the comment entries for the following comment entry statements are deleted.

DERIVATION(DER), NODERIVATION(NDER) Default: NODERIVATION

DESCRIPTION(DESC), NODESCRIPTION(NDESC) NODESCRIPTION

FALSE-WHILE(FW), NOFALSE-WHILE(NFW) NOFALSE-WHILE

PROCEDURE(PRCD), NOPROCEDURE(NPRCD) NOPROCEDURE

TRUE-WHILE(TW), NOTRUE-WHILE(NTW) NOTRUE-WHILE

VOLATILITY(VOL), NOVOLATILITY(NVOL) NOVOLATILITY

VOLATILITY-MEMBER(VOLM), NOVOLATILITY-MEMBER(NVOLM) NOVOLATILITY-MEMBER

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

VOLATILITY-SET(VOLS), NOVOLATILITY-SET(NVOLS)
NOVOLATILITY-SET

Output-Data PRINT, NOPRINT(NP) Default: PRINT

The PRINT parameter initiates the production of a printed DELETED COMMENT ENTRIES report. NOPRINT suppresses the printing.

Messages: If neither the FILE nor NAME parameter are given, the message:

NO NAME OR FILE SPECIFIED

is printed.

Examples: DELETE-COMMENT-ENTRY N=NEW-INFO-VALIDATION PRCD DESC

DCOM FILE DESC

Command: DELETE-PSL Type: modifier command

Purpose: To delete specific URL statements in the problem definer's data base. Those statements used as input to the command are deleted. A permanent record for the change is generated in the form of the DELETED URL report.

Prototype: DELETE-PSL(DPSL) [parameter]...

Parameters:

Input-Data INPUT(I)[=fdname] Default: INPUT=terminal

When INPUT is used and an fdname is specified, the contents of the designated fdname are used as input to the command. This input must be in the same format allowable by the INPUT-PSL command (i.e., legal URL statements). The only exception is that no comment entry statements are allowed in the input (DESCRIPTION, for example). The EOF statement terminates input. If no fdname is specified, its value defaults to the terminal so that the URL statements can be entered interactively.

Output-Data SOURCE(S), NOSOURCE(NS) Default: SOURCE

When the SOURCE parameter is in effect, an AS-IS SOURCE LISTING (the DELETED URL output) of the deleted URL statements is produced. When the NOSOURCE parameter is given, no AS-IS SOURCE LISTING is produced.

XREF(X), NOXREF(NX) Default: NOXREF

The user may desire a cross reference for the AS-IS SOURCE LISTING. This consists of a list of all user-defined names from the input file and the line numbers on which they occur in the DELETED URL report. To accomplish this, the problem definer should specify XREF. When NOXREF is in effect, no cross reference is produced.

Example:
```
DELETE-PSL  NS
DPSL   X
```

Command: DICTIONARY Type: report command

Purpose: To produce the DICTIONARY REPORT for a name or list of names in the user's data base

Prototype: DICTIONARY(DICT) [parameter]...

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name
Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for production of the DICTIONARY REPORT. When a name is specified via the NAME parameter, the report is generated for that name alone. The format of the input file must be one name per line.

Output-Data INDEX, NOINDEX Default: NOINDEX

When given, the INDEX parameter specifies the production of an index to the report. This index consists of an alphabetical listing of all names used in the report and the page(s) on which they occur in the report.

Output-Option The following four parameters specify the information t be included in the DICTIONARY. The "NO" prefix on a parameter specifies that such information not be included for the name(s).

DESCRIPTION(DESC),
NODESCRIPTION(NDESC) Default: DESCRIPTION

KEYWORDS(KEY), NOKEYWORDS(NKEY)
Default: KEYWORDS

RESPONSIBLE-PD(RPD),
NORESPONSIBLE-PD(NRPD) Default: RESPONSIBLE-PD

SYNONYMS(SYN), NOSYNONYMS(NSYN)
Default: SYNONYMS

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Output-Format

NUM-SPACE(NS)=integer Default: NUM-SPACE=2

For ease in reading, the number of lines skipped between dictionary entries can be modified by varying this parameter. NUM-SPACE may take on any value between 0 and 10.

Examples: DICTIONARY N=PAYROLL-PROCESSING

DICT FILE

Command: ENTITY-IDENTIFIER Type: report command

Purpose: To produce the IDENTIFIER INFORMATION REPORT.

Prototype: ENTITY-IDENTIFIER(EI) {IDENTIFIER(I)}
{ENTITY(E) } [parameter]...

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as in the input file for the command. If a name is specified via the NAME parameter, the report is generated for that name alone. The format of the input file must be one name per line.

Input-Control... IDENTIFIER(I), ENTITY(E) Default: no default

Since no default is allowed, one of the above must be specified. If IDENTIFIER is specified, the names used as input to the command must be names used as IDENTIFIERS in the data base. If the ENTITY parameter is given, the names used as input must be defined ENTITY names.

Messages: If neither the IDENTIFIER nor ENTITY parameter is specified, the message:

MUST GIVE EITHER ENTITY OR IDENTIFIER PARAMETER

will be given.

Examples: EI N=EMPLOYEE-NUMBER I

EI FILE ENTITY

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: EXTENDED-PICTURE Type: report command

Purpose: To produce the EXTENDED-PICTURE report.

Prototype: EXTENDED-PICTURE(EP) [parameter]...

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the comand. This file is the default PUNCH file for NAME-GEN. If an fdname is given, that file is used as the input file for the command. The format for the input file must be one name per line. When a name is given via the NAME parameter, the report is produced only for that name. Regardless of whether FILE or NAME is specified, all names used as input to this command must be PROCESS, INTERFACE, INPUT, ELEMENT, GROUP, OUTPUT, ENTITY, or SET names.

Output-Data INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the EXTENDED PICTURE report. This index consists of all user-defined names used in the report, in alphabetical order, along with the pages on which they appear in the report.

Output-Option STRUCTURE(STR), DATA-FLOW(DF) Default: no default

When the STRUCTURE parameter is used, information is obtained for each input name from UTILIZES, SUBPARTS, CONSISTS, and SUBSETS statements, or their complementary statements. This information is then presented in graphical format. When the DATA-FLOW parameter is used, the graphical output shows information obtained for each input name from USED, USED TO DERIVE, USED TO UPDATE, RECEIVED, UPDATES, DERIVES, and GENERATES statements or their complementary statements. Since there is no default, either STRUCTURE or DATA-FLOW must be specified.

FORWARD(FWD), BACKWARD(BWD) Default: no default
DOWNWARD(DOWN), UPWARD(UP)

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

It is frequently convenient to think of FORWARD and BACKWARD as referring to DATA-FLOW and UPWARD and DOWNWARD as referring to STRUCTURE. However, FORWARD and DOWNWARD may be used interchangeably, as may BACKWARD and UPWARD. The FORWARD (or DOWNWARD) parameter causes retrieval of information about each input name based on UTILIZES, SUBPARTS, CONSISTS and SUBSETS statements (for STRUCTURE) or USED, USED TO DERIVE, USED TO UPDATE, RECEIVED, UPDATES, DERIVES, and GENERATES statements (for DATA-FLOW). The BACKWARD (or UPWARD) parameter causes retrieval of information about each input name based on UTILIZED, PART, CONTAINED, and SUBSET statements (for STRUCTURE) or USES, USES TO DERIVE, USES TO UPDATE, RECEIVES, UPDATED, DERIVED, and GENERATED statements (for DATA-FLOW). Since there is no default, one of the above parameters must be specified.

LINKS=integer Default: LINKS=1000

The LINKS parameter specifies the maximum number of links (connections between names) to be followed in producing the report. LINKS can take on any integer value between 1 and 1000, inclusive.

Output-Format

COLUMNS(COLS)=integer Default: COLUMNS=119

The COLUMNS parameter specifies the number of columns to be used for output. The maximum value allowed is 119.

ROWS=integer Default: ROWS=39

The ROWS parameter specifies the number of rows to be used for output. The maximum value allowed is 39.

HORIZONTAL-BOXES(HB)=integer Default: (see text)

HORIZONTAL-BOXES specifies the maximum number of boxes containing names to be arranged across the page. The default value is the largest possible value for the given value of COLUMNS, and is computed as the greatest integer in (COLUMNS-4)/17.

VERTICAL-BOXES(VB)=integer Default: (see text)

VERTICAL-BOXES specifies the maximum number of boxes containing names to be arranged down the page. The default value is the largest possible value for the given value of ROWS, and is computed as the greatest integer in (ROWS-2)/6.

Messages: If the HORIZONTAL-BOXES value used will not fit in the number of COLUMNS specified, the message:

HORIZONTAL-BOXES TOO LARGE FOR NUMBER OF COLUMNS ON PAGE

will be given.

If the VERTICAL-BOXES value used will not fit in the number of ROWS specified, the message:

VERTICAL-BOXES TOO LARGE FOR NUMBER OF ROWS ON PAGE

will be given.

If neither the DATA-FLOW nor STRUCTURE parameter is specified, the message:

NEITHER DATA-FLOW NOR STRUCTURE WAS SPECIFIED

will be printed.

If none of FORWARD, BACKWARD, UPWARD and DOWNWARD is specified, one of the following messages will be printed:

NEITHER FORWARD NOR BACKWARD WAS SPECIFIED
NEITHER UPWARD NOR DOWNWARD WAS SPECIFIED.

Examples: EXTENDED-PICTURE STR DOWN N=PROCESS1
EP FILE DF BACKWARD LINKS=5

Command: FORMATTED-PROBLEM-STATEMENT Type: report command

Purpose: To produce the FORMATTED PROBLEM STATEMENT for a given name, or list of names and/or to produce this information in the form of PUNCH data.

Prototype: FORMATTED-PROBLEM-STATEMENT(FPS) [parameter]...

Parameters:

Input-Data: FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME=GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, the report is produced only for the name specified. The format of the input file must be one name per line.

Output-Data: INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the FPS. This index consists of an alphabetical listing of all user defined names used in the FORMATTED PROBLEM STATEMENT and the page(s) on which they occur.

PRINT, NOPRINT(NP) Default: PRINT

The NOPRINT parameter specifies that no printed output report will be produced. The PRINT parameter specifies the production of the FORMATTED PROBLEM STATEMENT.

PUNCH(P)[=fdname], NOPUNCH Default: NOPUNCH

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no fdname is designated, the data is written into the default PUNCH file.[1] This file is the default PUNCH file for the command. If an fdname is indicated, that file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Output- COMMENT(COM), NOCOMMENT(NCOM) Default: COMMENT

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Option

The COMMENT option, when in effect, specifies the inclusion of comments for undefined names. The NOCOMMENT option suppresses these comments.

DEFINE(DEF), NODEFINE(NDEF) Default: DEFINE

With the DEFINE option in effect, DEFINE sections are included in the report. The NODEFINE option specifies that no DEFINE sections are included in the FORMATTED PROBLEM STATEMENT.

DESG(DG), NODESG(NDG) Default: DESG

The DESG option, when in effect, indicates that DESIGNATE sections are provided for SYNONYM names in the FORMATTED PROBLEM STATEMENT. The NODESG option suppresses the production of such output.

EMPTY, NOEMPTY Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before PUNCH data is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

ALL-STATEMENTS(AS),NOALL-STATEMENTS(NAS)
Default: NOALL-STATEMENTS

The ALL-STATEMENTS option specifies that all statements will be printed for each section in the FORMATTED PROBLEM STATEMENT whether information was supplied or not.

LINE-NUMBERS(LNS),NOLINE-NUMBERS(NLNS)
Default: LINE-NUMBERS

Specifies whether or not line numbers are to be produced in the printed output.

PRINTEOF(PEOF),NOPRINTEOF(NPEOF) Default: PRINTEOF

Specifies whether an extra line containing EOF is to be produced at the end of the output.

COMPLEMENTARY-STATEMENTS(COMP)
Default: COMPLEMENTARY-
NOCOMPLEMENTARY-STATEMENTS(NCOMP)
STATEMENTS

Specifies whether complementary statements are to be produced. If the NOCOMPLEMENTARY-STATEMENTS option is in effect, then the number of statements and lines

in the output is the minimum which contains all the information in the data base.

Output-Format

AMARG(AM)=integer Default: AMARG=10

The AMARG parameter indicates the column at which the first name of a name pair is to be outputted. An example of a name pair can be found in the ATTRIBUTE statement where the syntax requires an ATTRIBUTE name and ATTRIBUTE-VALUE. AMARG must take on some value greater than SMARG and less than BMARG.

BMARG(BM)=integer Default: BMARG=25

The BMARG parameter indicates the column at which the second name of a pair is to be outputted. BMARG must take on some value greater than AMARG and less than RNMARG-30.

CMARG(CM)=integer Default: CMARG=1

The CMARG parameter specifies the number of columns from SMARG the text (comment entry) for a comment entry statement begins. CMARG must take on some value greater than 0 and less than RMARG-72.[1]

HMARG(HM)=integer Default: HMARG=40

This parameter specifies the column where the user defined name in a section header statement are to be printed on the output. HMARG must take on some value greater than SMARG and less than RMARG-30.[1]

NEW-LINE(NL), NONEW-LINE(NNL) Default: NONEW-LINE

When the NEW-LINE parameter is given, the first name of a name list associated with a statement will appear on the line succeeding the statement identifier (name of the statement). The NONEW-LINE parameter initiates the list on the same line as the statement identifier.

NEW-PAGE(NPG), NONEW-PAGE(NNPG)
Default: NONEW-PAGE

When given, the NEW-PAGE parameter specifies that each section of the FPS be printed on a separate page. NONEW-PAGE signifies that the sections will follow one another on a page within the page size restrictions. In any case, interrupted sections will be continued on succeeding pages.

---

[1] RMARG has the value 119 at most installations.

NMARG(NM)=integer Default: NMARG=20

The NMARG parameter indicates the column in which the name or first name of a name list for any statement will be outputted. NMARG must take on some value greater than SMARG and less than RNMARG-30.

ONE-PER-LINE(OPL), SEVERAL-PER-LINE(SPL)
Default: ONE-PER-LINE

The ONE-PER-LINE option indicates that the names in a name list for any statement will appear on succeeding lines. SEVERAL-PER-LINE option signifies that names in a name list will appear on the same line.

RNMARG(RM)=integer Default: RNMARG=70

Specifies the right-hand margin for names in a name list when the SEVERAL-PER-LINE parameter is in effect. RNMARG must take on some value greater than MAX(BMARG, NMARG)+29 and less than RMARG-30.[1]

SMARG(SM)=integer Default: SMARG=5

The SMARG parameter indicates the column in which the statement identifier (name of the statement) will be started. SMARG must take on some value greater than 0 and less than MIN(AMARG, NMARG).

Examples: FPS N=FIELD-CHECK-NEW

FPS FILE

---

[1] RMARG has the value 119 at most installations.

Command: FREQUENCY Type: report command

Purpose: To produce the FREQUENCY REPORT.

Prototype: FREQUENCY(FREQ)

Parameters:
None

Examples: FREQ

Command: HELP

Purpose: To provide the on-line user with a list of possible commands for URA or information about the parameters for a particular URA command.

Prototype: HELP [parameter]...

Parameters:

Command-name Default: (see text)

If no command-name is given, a list of currently available URA commands is given. If a command-name is given, then the parameters for that command are presented. Abbreviations for the command-name are also acceptable.

SHORT, LONG Default: SHORT

If SHORT is given, only the parameters for the given command are printed. If LONG is given, explanations of the various parameters are also printed.

Examples: HELP FPS LONG

HELP CONTENTS

Command: INPUT-PSL Type: modifier command

Purpose: To add information to the URA data base to expand or modify the problem statement. A permanent record of the change can be generated in the form of the AS-IS SOURCE LISTING and URA CROSS REFERENCE.

Prototype: INPUT-PSL(IP) [parameter]...

Parameters:

Input-Data INPUT(I)=fdname Default: INPUT=terminal

When INPUT is specified, the contents of the designated fdname are used as input to the command. This input must be in the format of legal URL statements as specified by Part II, URL User's Manual. The EOF statement terminates input. If the INPUT parameter is not specified, input is read from the terminal so that the URL statements can be entered interactively.

Input-Control DBREF(D), NODBREF(ND) Default: DBREF

The DBREF parameter allows the referencing of the data base by URA in its syntax and semantic analysis. When given, NODBREF allows the analyzer to only perform a syntax check of the input.

UPDATE(U), NOUPDATE(NU) Default: NOUPDATE

With the UPDATE parameter given, the input will update the URA data base. NOUPDATE indicates that the data base is not to be changed.

Output-Data SOURCE(S), NOSOURCE(NS) Default: SOURCE

When the SOURCE parameter is in effect, AS-IS SOURCE LISTING of the input URL is produced. When the NOSOURCE parameter is given, the listing is not produced.

XREF(X), NOXREF(NX) Default: NOXREF

XREF specifies that a URA CROSS REFERENCE is to be generated for the AS-IS SOURCE LISTING. This consists of a list of all user defined names from the input file and the line numbers on which they occur in the AS-IS SOURCE LISTING.

Examples: INPUT-PSL XREF UPDATE
IP U

Command: KWIC Type: report command

Purpose: To produce a KWIC INDEX for a list of names.

Prototype: KWIC [parameter]...

Parameters:

Input-Data FILE(F)[=fdname] Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the comand. The format of the input file must be one name per line.

Output-Format DIF=integer Default: DIF=20

DIF is the number of spaces allowed between the keyword and the rest of the name as it appears in the output. DIF must take on some value greater than 1 and less than 53.

Examples: KWIC DIF=10

KWIC

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: NAME-GENERATION Type: report command

Purpose: To produce the NAME-GENERATION report and/or retrieve certain names to be put in a PUNCH file and used as input to other commands.

Prototype: NAME-GENERATION(NG) [parameter]...

Parameters:

Output-Data

PRINT, NOPRINT(NP) Default: PRINT

The PRINT parameter initiates the production of the Name Generation Report: NOPRINT suppresses printing of the report.

PUNCH(P)[=fdname], NOPUNCH Default: PUNCH

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no fdname is designated, the data is written into the default PUNCH file.[1] This file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Output-Option

EMPTY, NOEMPTY Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before the list of names is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file. When PUNCH and NOEMPTY are specified together, the punched output appears at the end of the list of names already in the file.

SELECTIONS(S)='(boolean expression'),
INPUT(I)[=fdname] Default: no default

The SELECTION parameter is used to retrieve names of particular name types or which meet various selection criterion. The contents of the boolean expression specifies the types of names to be retrieved and how they are to be selected.

If the boolean expression (without primes) is stored in a file, the expression may be used to select names when the file is specified by the INPUT parameter.

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

A boolean expression consists of the following objects:

1. Operands

An explanation of each operand is given below.

NAME-TYPE

The following name types may be used as operands:

ATTRIBUTE	INTERFACE	REOUSRCE
ATTRIBUTE-VALUE	INTERVAL	RESOURCE-USAGE-PARAMETER
CLASSIFICATION	KEYWORD	SECURITY
CONDITION	MAILBOX	SET
ELEMENT	MEMO	SOURCE
ENTITY	OUTPUT	SUBSETTING-CRITERION
EVENT	PROBLEM-DEFINER	SYSTEM-PARAMETER
GROUP	PROCESS	UNDEFINED
INPUT	PROCESSOR	UNIT

ALL

When the ALL operand is specified, the names of all name types except SYNONYM and UNDEFINED will be presented.

TOTAL

When the TOTAL operand is specified, every name in the data base will be presented.

ATTR=attr-name[,value]

When the ATTR operand is specified, those names with the given user-name as an ATTRIBUTE are selected to be part of the output. The user-name must be a name defined as an ATTRIBUTE in the data base.

If an ATTRIBUTE-VALUE is specified as part of the user-name, then only those names with the given user-name as an ATTRIBUTE-VALUE, for the ATTRIBUTE designated by the ATTR parameter, are selected to be part of the output. The user-name must be an ATTRIBUTE-VALUE name in the data bse.

SUBPARTS-OF(SO)=user-name[,level]

All names which belong to the SUBPARTS structure for a

AD-A057 130 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS ANALYZER (URA) USER'S MANUAL H6180/MULTICS/VE--ETC(U)
MAR 77 D TEICHROEW, A HERSHEY, S SPEWAK F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-128 NL

5 OF 6
AD A057130

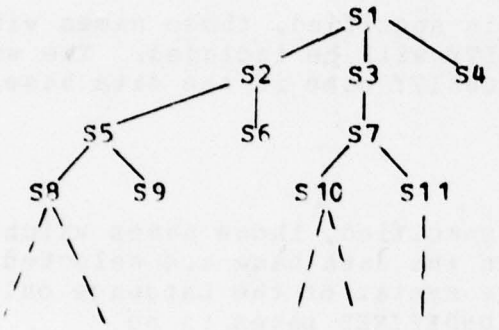


given name (as would be retrieved for the STRUCTURE report) can be retrieved by specifying:

SUBPARTS-OF=name

where the name is an INPUT, OUTPUT, PROCESS or INTERFACE name which has SUBPARTS information defined for it. The number of levels to go down and retrieve names to present in the report is specified by the SUBLEVEL parameter or by attaching a command a level number after the user-name with the SUBPARTS-OF parameter.

If SUBLEVEL=0, then all levels of names are presented. If SUBLEVEL=1, then only those names which are PART OF the SUBPARTS-OF name are presented. The following picture may clarify the association between the value of SUBLEVEL and the names presented.



Generation of the report with SUBPARTS-OF=S1 and SUBLEVEL=3 would present S2, S3, S4, S5, S6, S7, S8, S9, S10 and S11 in the report. Generation of the report with SUBPARTS-OF=S1 and SUBLEVEL=1 would present the names S2, S3 and S4.

SYNONYMS

When the SYNONYMS operand is specified, all SYNONYMS are presented for each name retrieved in the report in addition to the basic form of the name. If only the SYNONYMS are desired, the basic names may be suppressed by specifying the NOBASIC and SYNONYM parameters. With standard defaults in effect, the BASIC and NOSYNONYM parameters are used.

KEY=user-name

When the KEY operand is specified, those names with the given user-name as a KEYWORD are selected to be part of the output. The user-name must be a name defined as a KEYWORD in the data base.

PD=user-name

When the PD operand is specified those names with the given user-name as a KEYWORD are selected to be part of the output. The user-name must be a name defined as a KEYWORD in the data base.

SOURCE=user-name

When the SOURCE operand is specified, those names with given user-name as a SOURCE will be included in the output. The user-name must be defined as a SOURCE in the data base.

SECURITY=user-name

When the SECURITY operand is specified, those names with given user-name as a SECURITY will be included. The user-name must be defined as SECURITY name in the data base.

USAGE=ID, IDENTIFIER

When the USAGE operand is specified, those names which have user-name as IDENTIFIERS in the data base are selected to be part of the output. The syntax of the Language only allows ELEMENT, GROUP and UNDEFINED names to be IDENTIFIERS.

2. Operators

An explanation of each operator is given below.

NOT, ¬

The NOT operator placed before an operand specifies that the names associated with that operand will not be retrieved. For example, a boolean expression of the form NOT PROCESS means that all names in the data base except for PROCESS names will be retrieved.

AND,& ,*

The AND operator specifies that any name retrieved from the data base must meet the criterion designated before and after the AND operator. For example, a boolean expression of the form PROCESS AND KEY=level-1 means that any name retrieved must be a PROCESS name as well as have the KEYWORD "level-1" attached to it.

OR,|,+

The OR operator specifies that any name retrieved from the data base must either meet the criterion designated before the OR operator, or after the operator, or both. For example, a boolean expression of the form PROCESS OR KEY=level-1 means that any name retrieved must be either a PROCESS name, or have the KEYWORD "level-1", or be both a PROCESS and have the KEYWORD "level-1".

3. Other Notations

{ }

Braces may be used to group operands and operators in a boolean expression.

The following rules apply to the formation of boolean expressions to be used with the SELECTION parameter of the NAME-GEN command.

The following notation is used to define a boolean expression:

exp - boolean expression
opd - operand
opr - operator

1) A boolean expression (exp) can only be in one of the following forms:

a) opd
b)[1] ¬ exp
c) (exp)
d) exp opr exp

2) A boolean operator (opr) must be one of the following (equivalent operators are listed in parentheses):

a) & (AND,*)
b) | (OR,+)

3) A boolean operand (opd) must be one of the allowable options for the SELECTION parameter as listed below:

a) name-type
b) KEY=user-name
c) PD=user-name
d) SOURCE=user-name

---

[1] The form "¬ exp" can also be written as "not exp" or "¬exp".

e) SECURITY=user-name
f) USAGE=user-name
g) ATTR=user-name
h) ATTR=user-name,user-name
i) ATTR=user-name,integer
j) SUBPARTS-OF(SO)=user-name
k) SUBPARTS-OF(SO)=user-name,ALL
l) SUBPARTS-OF(SO)=user-name,integer
m) SUBLEVEL(SL)={ALL|integer}

A name-type option can be any of the following:

ATTRIBUTE(ATTR)	PROCESSOR(PRCR,PROCR)
ATTRIBUTE-VALUE(ATTV)	REAL-WORLD-ENTITY(RWE)
CLASSIFICATION(CLS)	RELATION(RLN)
CONDITION(COND)	RESOURCE(RSC)
ELEMENT (ELE)	RESOURCE-USAGE-PARAMETER(RUP)
ENTITY(ENT)	SECURITY(SEC)
EVENT(EV)	SET
GROUP (GR)	SOURCE(SRC)
INPUT(INP)	SUBSETTING-CRITERION(SSCN)
INTERFACE(INTF)	SYSTEM-PARAMETER(SYSP)
INTERVAL(INT)	UNDEFINED
KEYWORD(KEY)	UNIT
MAILBOX(BOX)	ALL
MEMO	TOTAL
OUTPUT(OUT)	BASIC
PROBLEM-DEFINER(PD)	SYNONYM(SYN)
PROCESS(PROC)	

The rules which must be followed in specifying these boolean expressions are as follows:

1) Any number of blanks in a boolean expression may occur:

a) on either side of an operand
b) before and after the symbol "¬"
c) before and after the symbols "(" and ")"
d) on either side of an operator

2) No blanks may occur within an operator and no substitutes, other than the specified symbols are allowed to represent operators.

3) No blanks may occur within an operand and no operands other than the specified operands may be used.

4) Assume that AND has precedence over OR when parentheses are not given. For example, the expression ELEMENT AND GROUP OR KEY=low is interpretted as (ELEMENT AND GROUP) OR KEY=low rather than ELEMENT AND (GROUP OR KEY=low).

5) A "¬" is assumed to belong to the immediately following operand unless specified otherwise. For example, the

expression ¬ PROCESS OR ELEMENT would be interpretted as {¬ PROCESS} OR ELEMENT rather than ¬ { PROCESS OR ELEMENT}.

6) A boolean expression may be continued over several lines when the last character on the line is a dash (-), when it appears in the ordinary command stream. When INPUT= is specified, the boolean expression must appear in card image format, with no operands broken between lines and no dashes for continuation.

The following boolean expressions are illegal for the NAME-GEN command:

1) PROCESS OR { GROUP KEYWORD=level-1}

No operator between operands GROUP and KEYWORD

The following boolean expressions are legal:

1) ELEMENT OR GROUP OR ENTITY

2) PROCESS AND KEYWORD=S1 OR KEYWORD=level-1

3) { GROUP | ELEMENT } AND ATTR=type,character

Output-Format

ORDER={BYTYPE|ALPHA|attr-name}...
Default: ORDER=BYTYPE

The ORDER parameter specifies how the names retrieved are to be ordered (in the PUNCH file and the report).

When ORDER=ALPHA, all names retrieved are presented in alphabetical order by name. When ORDER=BYTYPE, all names retrieved are grouped by the name types associated with them. The name types are ordered alphabetically as are the names within each name type group. When ORDER=attr-name or ORDER=attr-name-synonym, all names retrieved are presented in alphabetical order of the values they have for the specified ATTRIBUTE name. Names which do not have the specified ATTRIBUTE are placed at the beginning of the list of names.

In addition to the three different orderings that may be specified, an ordering list may be given which allows up to five levels of ordering on the retrieved names. Only the BYTYPE option and ATTRIBUTE names may appear in the ordering list. (The ALPHA option may not.) The BYTYPE option may only appear once in the list. All options in the list must be separated by commas. No blanks may appear in the list. The names are ordered first on the first option in the

ordering list; within that ordering, on the second option in the ordering list, etc. Some examples of using the ordering list are:

```
ORDER=BYTYPE,length,type
ORDER=level,BYTYPE
```

Examples

```
NG S='PROCESS AND KEY=level-2'

NG S='(NOT PROCESS) * KEY=level-2'

NG S='ATTR-occurrence, unscheduled'

AND SO ='security-control-input'

NG S='SO=security-control-input,1'

ORDER=BYTYPE, occurrences
```

Command: NAME-LIST Type: report command

Purpose: To produce the NAME LIST report.

Prototype: NAME-LIST (NL) [parameter]...

Parameters:

Output-Format

ORDER={ALPHA|BYTYPE} Default: ORDER=ALPHA

If ORDER=ALPHA is specified, the list is ordered by the URA name of the object, if ORDER=BYTYPE is specified, the order is alphabetical by name type, with objects of the same type being ordered by name.

Examples: NAME-LIST

NL ORDER=BYTYPE

Command: PICTURE Type: report command

Purpose: To produce the PICTURE report.

Prototype: PICTURE(PIC) [parameter]...

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, the report is produced only for that name. In any case, the names used as input to this command must be INTERFACE, PROCESS, SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. The format of the input file must be one name per line.

Output-Data INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the PICTURE report. This index consists of all user defined names used in the report, in alphabetical order and the pages on which they appear in the report.

Output-Option DATA(D), NODATA(ND) Default: DATA

With the DATA parameter in effect, information applicable and available for the given name, or list of names other than structure or flow data is printed on the output. NODATA inhibits such action.

FLOW, NOFLOW Default: FLOW

This parameter presents flow information in the PICTURE report. It presents RECEIVES and GENERATES information between INPUTS and OUTPUTS with PROCESSES and INTERFACES. It also presents USES and DERIVES information between PROCESSES and data (such as SETS, ENTITIES, GROUPS and ELEMENTS).

STRUCTURE(STR), NOSTRUCTURE(NSTR) Default: STRUCTURE

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

When the STRUCTURE parameter is in effect, the information available in the SUBPARTS, CONSISTS and/or SUBSETS statements and their complementary statements for the input name(s) appears in the report.

Examples: 'PICTURE N=PAYCALC-UPDATING

PIC N=PAYROLL-PROCESSING NODATA NOFLOW

Command: PRINT-ATTRIBUTE-VALUES Type: report command

Purpose: To produce the ATTRIBUTE REPORT.

Prototype: PRINT-ATTRIBUTE-VALUES(PAV) [parameter]...

Parameters:

Input-Data FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The input file format must be one ATTRIBUTE name per line. When a name is specified by the NAME parameter, the report is generated for that name only. In any case, only ATTRIBUTE names may be used as input to this command.

Examples: PRINT-ATTRIBUTE-VALUES FILE

PAV N=TYPE

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: PROCESS-CHAIN Type: report command

Purpose: To produce the PROCESS CHAIN report.

Prototype: PROCESS-CHAIN(PC) [parameter]...

Parameters:

Input-Data — FILE(F)[=fdname],NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is given, that file is used as the input file for the command. The format for the input file must be one name per line. When a name is given via the NAME parameter, the report is produced only for that name. Regardless of whether FILE or NAME is specified, all names used as input to this command must be EVENT or PROCESS names.

Output-Data — INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the PROCESS CHAIN report. This index consists of all user-defined names used in the report, in alphabetical order, along with the pages on which they appear in the report.

Output-Option — LINKS=integer Default: LINKS=1000

The LINKS parameter specifies the maximum number of links (connections between names) to be followed in producing the report. LINKS can take on any integer value between 1 and 1000, inclusive.

Output-Format — COLUMNS(COLS)=integer Default: COLUMNS=119

The COLUMNS parameter specifies the number of columns to be used for output. The maximum value allowed is 119.

ROWS=integer Default: ROWS=39

The ROWS parameter specifies the number of rows to be used for output. The maximum value allowed is 39.

HORIZONTAL-BOXES(HB)=integer Default: (see text)

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

HORIZONTAL-BOXES specifies the maximum number of boxes containing names to be arranged across the page. The default value is the largest possible value for the given value of COLUMNS, and is computed as the greatest integer in (COLUMNS-4)/17.

VERTICAL-BOXES(VB)=integer Default: (see text)

VERTICAL-BOXES specifies the maximum number of boxes containing names to be arranged down the page. The default values is the largest possible value for the given value of ROWS, and is computed as the greatest integer in (ROWS-2)/6.

Messages: If the HORIZONTAL-BOXES value used will not fit in the number of COLUMNS specified, the message:

HORIZONTAL-BOXES TOO LARGE FOR NUMBER OF COLUMNS ON PAGE

will be given.

If the VERTICAL-BOXES value used will not fit in the number of ROWS specified, the message:

VERTICAL-BOXES TOO LARGE FOR NUMBER OF ROWS ON PAGE

will be given.

Examples: PROCESS-CHAIN N=EVENT1

PC FILE LINKS=4 INDEX

Command: PROCESS-INPUT-OUTPUT Type: report command

Purpose: To produce the PROCESS INPUT/OUTPUT report.

Prototype: PROCESS-INPUT-OUTPUT(PRIO) [parameter]...

Parameters:

Input-Data FILE(F)=[fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. When a single name is specified by the NAME parameter, the report is produced for that name alone. Either FILE or NAME can be used but not both. In any case, all the names used as input to this command must be PROCESS names. The input file format is one PROCESS name per line.

Output-Data INDEX, NOINDEX Default: NOINDEX

When given, the INDEX parameter specifies the production of an index into the report. The index consists of all input and output names in the report, in alphabetical order and the page(s) on which they occur in the report.

PRINT, NOPRINT(NP) Default: PRINT

The NOPRINT parameter specifies that no printed output report will be produced. The PRINT parameter specifies the production of the PROCESS INPUT/OUTPUT report.

Output-Option DESCRIPTION(DESC), NODESCRIPTION(NDESC) Default: DESCRIPTION

When the DESCRIPTION parameter is in effect, the comment-entry associated with the DESCRIPTION statement, for all PROCESS used as input, is retrieved and printed on the report. NODESCRIPTION specifies that this information is not to be retrieved.

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

PROCEDURE(PRCD), NOPROCEDURE(NPRCD)
Default: NOPROCEDURE

When the PROCEDURE parameter is specified, the comment entry associated with the PROCEDURE statement, for each PROCESS name used as input, is retrieved and printed on the report. With the NOPROCEDURE parameter in effect, this information is not retrieved.

INPUT(INP), NOINPUT(NINP) Default: INPUT

When the INPUT parameter is in effect, all the names of objects used as input to each PROCESS (i.e., names associated with the RECEIVES and USES statements) are retrieved and printed on the report. The NOINPUT parameter specifies that this information is not to be retrieved.

OUTPUT(OUT), NOOUTPUT(NOUT) Default: OUTPUT

When the OUTPUT parameter is in effect, all the names of objects designated as output from each PROCESS (i.e., names associated with the GENERATES, DERIVES, and UPDATES statements) are retrieved and printed on the report. The NOOUTPUT parameter specifies that this information is not to be retrieved.

Output-Format NEW-PAGE(NPG), NONEW-PAGE(NNPG)
Default: NONEW-PAGE

When given, the NEW-PAGE parameter specifies that each section of the PROCESS INPUT/OUTPUT report be printed on a separate page. NONEW-PAGE signifies that the sections will follow one another on a page within the page size restrictions. In any case, interrupted sections will be continued on succeeding pages.

Examples: PRIO N=PAYROLL-PROCESSING

PRIO F NDESC NPG PRCD

Command: PUNCH-COMMENT-ENTRY Type: report command

Purpose: To produce the PUNCHED-COMMENT-ENTRIES report and/or punch the specified comment entries into a PUNCH file.

Prototype: PUNCH-COMMENT-ENTRY(PCOM) [parameter]...

Parameter:

Input-Data FILE(F)[=fdname], NAME(N)=user-name
Default: FILE
When the FILE parameter is used and no fdname is designated, the contents of the default file[1] are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is specified by the NAME parameter, the output is produced for that name alone. The format of the input file must be one name per line.

Output-Data PRINT, NOPRINT(NP) Default: PRINT

The PRINT parameter initiates the production of printed output for the report. When the NOPRINT parameter is given, the PUNCH COMMENT ENTRIES report is not produced.

PUNCH(P)[=fdname], NOPUNCH Default: PUNCH

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no fdname is designated, the data is written into the default PUNCH file.[1] This file is the default PUNCH file for the command. If an fdname is indicated, that file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Output-Option EMPTY, NOEMPTY Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before PUNCH data is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

The comment entries associated with the following

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

types of comment entry statements are retrieved when given as parameters.

	Default:
DERIVATION(DER), NODERIVATION(NDER)	NODERIVATION
DESCRIPTION(DESC), NODESCRIPTION(NDESC)	NODESCRIPTION
FALSE-WHILE(FW), NOFALSE-WHILE(NFW)	NOFALSE-WHILE
PROCEDURE(PRCD), NOPROCEDURE(NPRCD)	NOPROCEDURE
TRUE-WHILE(TW), NOTRUE-WHILE(NTW)	NOTRUE-WHILE
VOLATILITY(VOL), NOVOLATILITY(NVOL)	NOVOLATILITY
VOLATILITY-MEMBER(VOLM), NOVOLATILITY-MEMBER(NVOLM)	NOVOLATILITY-MEMBER
VOLATILITY-SET(VOLS), NOVOLATILITY-SET(NVOLS)	NOVOLATILITY-SET

Examples: PCOM N=PAYROLL-PROCESSING DESC

PCOM F DESC PRCD

Command: RENAME Type: modifier command

Purpose: To change the name of some object in the data base and to produce the RENAME REPORT as a permanent record of the change.

Prototype: RENAME(REN) {OLD(O)=user-name NEW(N)=user-name}
{INPUT(I)=fdname }

Parameters:

Input-Data INPUT(I)=fdname Default: no default

For multiple name changes, an input file can be used. Each line of the file must consist of the old name followed by the new name. The two names must be separated by one or more blanks.

OLD(O)=user-name Default: no default

The user-name specified here is the name that is to be changed. This name must be defined in the data base.

NEW(N)=user-name Default: no default

The user-name specified here is the name to replace the old name. If the new name is already in the data base, the name will not be changed.

For a single change, both OLD and NEW must be given with legal values.

Messages: If neither INPUT nor the OLD and NEW parameters are specified the message:

MUST GIVE OLD AND NEW, OR INPUT

will be given.

Examples: RENAME OLD=EMPLOYEE-CODE NEW=EMPLOYEE-NUMBER

Command: REPLACE-COMMENT-ENTRY Type: modifier command

Purpose: To replace, for a given name, specific comment entries associated to it. A REPLACED COMMENT ENTRIES report is also printed as a permanent record of the change.

Prototype: REPLACE-COMMENT-ENTRY(RCOM) [parameter]...

Parameters:

Input-Data INPUT(I)=fdname Default: (see text)

The designated fdname contains the new comment entries that will replace specified old comment entries in the data base. The required format of the file is the same as that punched by PUNCH-COMMENT-ENTRY. If INPUT is not given, the input will be taken from the default file.[1] This file is the default PUNCH file for PCOM. For each comment entry to be replaced, the following format must be given in the input file:

```
name

comment-entry type;
   .
   .
   .
comment entry text
   .
   .
   .                    ;
```

Where name is a name defined in the data base, the comment-entry-type (e.g., DESCRIPTION, VOLATILITY, etc.) must be followed by a semicolon. The text following this must also be followed by a semicolon. This sequence of lines can be repeated as many times as necessary in the input file.

Output-Data PRINT, NOPRINT(NOP) Default: PRINT

The PRINT parameter initiates the production of the REPLACED COMMENT ENTRIES report; NOPRINT suppresses printing. The report, if produced, contains both the old and new comment entries.

Examples: RCOM NOPRINT

---

[1] The name of the default file is installation dependent and consequently is given in Appendix E.

Command: RESOURCE-CONSUMPTION-ANALYSIS Type: report command

Purpose: To produce the RESOURCE-COMSUMPTION REPORT

Prototype: RESOURCE-CONSUMPTION-ANALYSIS(RCA)
INTERVAL=user-name [parameters]

Parameters:

Input-Data FILE(F)[=fdname],NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. When a single name is specified by the NAME parameter, the report is produced for that name alone. Either FILE or NAME can be used but not both. In any case, all the names used as input to this command must be PROCESS names. The input file format is one PROCESS name per line. The given process-name is used as the name of the root process from which all subsequent analysis proceeds. The report will be produced only for consumption of resources in performing this process and its component processes as determined by the PROCESS-LEVEL parameter.

Analysis-Control PROCESS-LEVEL(PL)={ALL } Default: ALL
{number}

This parameter specifies the lowest level of the component PROCESSES relative to the root name whose information is used in the analysis. In short, it corresponds to the "depth" of UTILIZES and SUBPARTS connections that will be probed from the root-process. If ALL is in effect, probing continues until the lowest level processes are reached.

INTERVAL(INT)=user-name Default: none

This specifies the interval in terms of which the analysis and report production is done. The user-name must be an INTERVAL name. All necessary HAPPENS statements that are used by the Analyzer must be in terms of this INTERVAL, or must be convertible to it (through CONSISTS OF statements in the INTERVAL section).

Output-Data BYPROCESSOR(BP),NOBYPROCESSOR(NBP)
Default: BYPROCESSOR

When the BYPROCESSOR parameter is in effect, the Resource Consumption Report arranged by PROCESSOR

name is presented. When NOBYPROCESSOR is in effect, it is not presented.

BYRESOURCE(BR),NOBYRESOURCE(NBR)
Default: BYRESOURCE

When the BYRESOURCE parameter is in effect, the Resource Consumption Report arranged by RESOURCE name is presented. When NOBYRESOURCE is in effect, it is not presented.

INDEX,NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the Resource Consumption Report. This index consists of an alphabetical listing of all URL user-names included in the report and the page(s) on which they occur.

PROCESSOR-KEYWORD(PK)=
{ALL } Default: ALL
{user-name}

This parameter specifies that only those processors with a given key will have processor-resource utilization reports produced. The user-name must be a KEYWORD name. If ALL is in effect, all PROCESSORs that are used to perform the given process will have this report produced.

Example: RESOURCE-CONSUMPTION-ANALYSIS

N=MAIN-PROCESSING INTERVAL=YEAR

RCA INTERVAL=MONTH INDEX PL=1 PK=HARDWARE

Command: SECURITY-ANALYSIS Type: report command

Purpose: To produce the SECURITY ANALYSIS report.

Prototype: SECURITY-ANALYSIS(SECA) [parameters]

Parameters:

Input-Data FILE(F) [=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. When a single name is specified by the NAME parameter, the report is produced for that name alone. Either FILE or NAME can be used but not both. In any case, all the names must be data types (SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS, ELEMENTS) or types which potentially access data types (PROCESSES, INTERFACES, PROCESSORS). The input file format is one user name per line. The report will be produced for each name in the input file.

Analysis-Control PRINT-NULL-SECURITY-INFORMATION(PNSI)
NOPRINT-NULL-SECURITY-INFORMTION(NPNSI)
Default: NPNSI

If this parameter is specified, a list is produced of names in the data base for which security information could be defined but for which security information has not been defined.

PRINT-MATRIX(PMAT), NOPRINT-MATRIX(NPMAT)
Default: PMAT

If this parameter is specified the security conflicts matrix is printed. Otherwise, the matrix is not printed.

Output-Data INDEX, NOINDEX Default: NOINDEX

The index parameter specifies the production of an index for the Security Analysis Report. This index consists of an alphabetical listing of all user-names included in the report and the page(s) on which they occur.

Examples: SECURITY-ANALYSIS NAME=hourly-employee-processing PNSI

SECA INDEX

Command: STRUCTURE Type: report command

Purpose: To produce a STRUCTURE report for INPUTS, OUTPUTS, PROCESSES, INTERFACES or PROCESSORS.

Prototype: STRUCTURE(STR) [parameter]...

Parameters:

Output-Data INDEX, NOINDEX Default: NOINDEX

The INDEX parameter, when given, specifies the production of an index to the report, giving the pages on which each undefined name used in the report occurs. NOINDEX specifies that no index should be generated.

Output-Option

URA will produce structure reports for the following name types when given as parameters. (Only one may be given for each report.)

INPUT(INP) Default: PROCESS
OUTPUT(OUT)
PROCESS (PROC)
INTERFACE(INTF)[1]

PROCESSOR(PROR)

Output-Format INDENT(IND)=integer Default: INDENT=3

The number is the number of spaces to indent each succeeding level in the report. INDENT must take on some value greater than 0 and less than 11.

Examples: STRUCTURE
STR INPUT

---

[1] REAL-WORLD-ENTITY (RWE) may be used in place of INTERFACE.

Command: SUMMARY Default: report command

Purpose: To produce the DATA BASE SUMMARY output.

Prototype: SUMMARY(SUM)

Parameters:
no parameters

Examples: SUMMARY

SUM

PART V

AUTOMATED DOCUMENTATION SYSTEM

USER'S MANUAL

INTRODUCTION

Whenever an organization develops an information system, one of the major problems is maintaining it over its life. The developers of the system are rarely around, or available to help the maintenance group in their task. A frequent result of this situation is a complete redesign and redevelopment of the system to make relatively minor modifications to the system. Such practice is an obvious waste of resources that could have been avoided with proper documentation. While the above is a strong motivator for the need for documentation standards, there are other advantages involved, namely, better communication between developers of the system, easier training of new employees, etc. Hence, most organizations have developed their own standards which must be adhered to for documenting the system. Examples of such standards are MILITARY standards 483 and 490 and Department of Defense Manual 4120.17-M.

Much of the information that is needed in the final document describing the system may be obtained during the system's analysis, design, and construction phases of the development life cycle. Therefore, what is needed is a mechanism that will allow the capture of the information as and when it becomes available and storing it in a readily usable data base. This is where the URL/URA data base comes into the picture.

The Automated Documentation System provides the means of easily producing a document according to a specific standard. The process of creating the document is a three step procedure. A table of contents or structured outline for the desired document must be completed. This information must be stated in a prescribed format (see Section 1). This part of the process is referred to as the creation of the documentation schema. The Analyzer data base is used to store and easily keep the information about the system being documented current. The Analyzer data base is created by an analyst or documentor who is concerned with capturing an up-to-date and complete description of the system. There is no ordering to these steps. The analyst or documentor must insure that all information required by the documentation standard is in the Analyzer data base. The third part of the documentation procedure is to describe the document body in a prescribed format (see Section 2). The process is referred to as the creation of the documentation source. Consistency and coordination between the documentation schema, documentation source and Analyzer data base is required to automatically generate good document.

The three components of the Automatic Documentation System are the documentation schema, the documentation source and the Analyzer. These areas are discussed in this Part to clarify the usage of these terms and other terms in this Part, a glossary is included.

1.0 Document Initialization

The initialization for the Automated Documentation System must be performed for each project to be documented. This task may be performed by the project leader. It involves taking the Military Standard and deriving from them relevant headings for the particular project in mind. For this paper Military Standard 4120.17-M will be used as the basis (template) for the generation of the document. Portions of this document are shown in the example (Appendix B). The initialization allows the project director to identify the major and detailed areas of documentation to be produced by the project team members. It can be most succinctly expressed as the table of contents to be shown in the final documentation. Hence, the documentation schema is equivalent to a listing of the entries in the table of contents or structured outline of the desired document. The details of the syntax will be discussed in Section 1.2. Development of a documentation schema is discussed in Section 4.1. Section 1.1 will discuss some of the strategy to be followed, and the general initialization process. The end result of the initialization process is a documentation object schema.

1.1 The Documentation Schema - Strategy for the Document Initialization

There are basically three types of entries that may be entered into the documentation schema. Any entry line is a maximum of 80 characters. The entries may be typed in on cards, or other media; in this form they are known as documentation source schema. When these same entries have been entered on a file on the target computer which also has the Analyzer data base on file, the entries are know as the documentation object schema. The process of converting the source schema into the object schema would generally involve nothing more than reading in the cards, etc., into a file. The source schema is external to the computer while the object schema is stored in a file. One of the most common ways of doing this is by a local text editor, or some other file populating mechanism. This processor (editor, etc.) is known as the documentation initializer (Figure 67).

The three types of entries that may be entered into the documentation schema are:

- the table of contents (SECTION ENTRIES)
- formatting commands (TEXT FORMATTING ENTRIES)
- comments (COMMENT ENTRIES)

The table of contents entries are all preceded by a distinguishing character, "*" which must be entered on every heading to appear in the table of contents. Formatting commands are proceeded by the distinguishing character "*".

Finally, the manager may want to insert, into the documentation schema, personal comments which are not to appear in the final table of contents. These may be specified by a "&" followed by a comment of up to 79 characters. Should the comment be longer, it may be continued on the next line but preceded by another "&". Comments may be used to guide the documentors, to assign specific portions of the documentation to particular individuals, etc. Any formatting command that appears in the schema will be executed directly before the section is printed in the document. This characteristic will be discussed in Section 4.1.

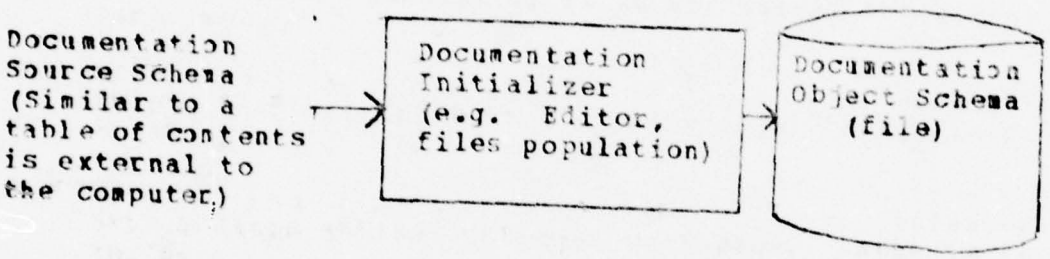

Documentation
Source Schema
(Similar to a
table of contents
is external to
the computer)
Documentation
Initializer
(e.g. Editor,
files population)
Documentation
Object Schema
(file)

Figure 67

1.2 Syntax for Entries in the Documentation Schema

SECTION-ENTRIES have the following syntax:

#SECTION-IDENTIFIER [HEADING-ENTRY]

The section-identifier must be preceded by a "#" in column 1. The section-identifier is the unique string corresponding to that section of the documentation schema (e.g., 2.2.3). This string may be made up of numbers, characters and/or special characters. No particular restrictions are placed on the format of the identifier (except for those placed or by a specific installation's control characters); for example, an identifier of 1.2s.1.b is a perfectly legitimate identifier. The section entry may not exceed 80 characters. It is important to note that there must be at least one space between the identifer and the heading entry. The final documentation will be produced in the order that the identifiers were entered in the documentation schema. Hence, it is the responsiblity of the individual preparing the documentation schema to enter them in the proper order. The documentation generator will not check to see that these are sorted. The heading-entry is the title to be given to the section. This is the title that will appear in the final document.

TEXT-FORMATTING-ENTRIES have the following syntax:

*CONTROL-INFORMATION-ENTRY

The text formatting commands may also be interspersed anywhere in the schema, but must be preceded by a "*" in column 1. There are two functions of formatting commands. A command can control what is going to be done at a certain point in regards to spacing or control the global options of formatting.

The following options can be used:

*NEW-PAGE
Skip to the top of the next page.

*SKIP [N] Default: N=1
Skip N lines. If N is greater than the number of lines left on a page, then this command causes carriages to skip to the top of the next page.

*HOLD-BLANK [N] Default: N=1
Check to see if N lines are left on this page. If there are, skip N lines. If there are not, skip to the next page and then skip N lines. This option insures that enough space will be left on a single page for a diagram or table to be written in manually.

*HOLD[N] Default: N=1
The rest of the current page is checked to see if N lines

are left on the page (not including HEADING or BOTTOM-LINES). If N lines are left, the next line of the source data base is produced. If N lines are not left on the current page, skip to the top of the next page and produce the next line of source.

These options are present at the commencement (under their default conditions and can be changed throughout the source):

*TITLE {OFF} Default: OFF
{Title}
If a title is given, this title is put on the top of each page. If OFF is given, no title is printed on each page.

*HEADING {OFF} Default: ON
{ON}
If ON is specified, the HEADING-ENTRY for the last section printed is put at the bottom of each page. If OFF is specified, the HEADING-ENTRY is not put at the bottom of each page.

*TOP-LINES [N] Default: N=3
Skip N lines at the bottom of each page. Range is from 0 to 10.

*BOTTOM-LINES [N] Default: N=2
Skip N lines at the bottom of each page. Range is from 0 to 10.

*LINES [N] Default: N=60
N is the number of lines to be printed on a page, including Title and Heading. Range is from 5 to 60.

*SOURCE-MARGIN [N] Default: 5
N is the number of spaces to indent the actual text material. This does not affect Analyzer report indentation. Range is from 1 to 40.

HEADING-MARGIN [N] Default: N=10
N is the number of spaces to indent a section heading. Range is from 1 to 40.

*HEADING-SKIP [N] Default N=2
N is the number of lines to skip before printing a section-header. The amount skipped will at most be to the top of the next page. Range is from 0 to 60.

*REPORT-CC {OFF} Default: ON
{ON}
When set to ON, the Analyzer report carriage control is engaged when outputting Analyzer reports. This affects page feeding of the document produced. When OFF, the line skipping and page feeding is controlled by only formatting commands.

COMMENT-ENTRIES have the following syntax:

```
& COMMENT
```

Comments may be interspersed anywhere in the documentation schema. They will not appear in the final output. The "&" must appear in column 1. The comment on a given line can be a maximum of 79 characters. There is no limit to the number of comment entries and header entries in a documentation source schema. An example of a documentation source schema is:

```
#2.  PROJECT DEVELOPMENT ENVIRONMENT/DOCUMENTATION SYSTEM

#2.1 DEVELOPMENTAL PHASES

#2.1.1 INITIATION

&THERE SHOULD BE NO NEED TO HAVE ANY OTHER LISTS FOR US.
```

(A larger example is in Appendix F.)

2.0 Documentation Source Population

The actual documentation to be written would probably be done by more than one individual. These individuals may enter the documentation of the areas they are responsible for, into different files, or into one common file assigned to be shared by them. In the former case, the documentation manager must merge the different pieces of the documentation into a single file. The order of this file is not important, as the final documentation will be produced in the order specified by the documentation schema (see Section 1). The input for the documentation source is referred to as the documentation source input.

The documentation source contains the information on the contents of the document body. This information is expressed by various types of entries. There are five entry types: Section-entries, analyzer-command-entries, text-formatting-entries, comment-entries and text entries.

2.1 Strategy for the Documentation Source Population

It is important that the individuals who are going to document the system know what is in the Analyzer data base. They should be given a complete NAME-LIST[1] (or NAME-GEN 1) of names in the Analyzer data base. They should have a FORMATTED-PROBLEM-STATEMENT 1 of all these names, or have access via a terminal, to the Analyzer data base to query the information within it. For example, if the documentation calls for the description of a particular input (say TEST-RESULTS), this should be in the Analyzer data base, and the documentor should be required to invoke its description from the Analyzer data base rather than to have to retype it. This has two obvious advantages: the one mentioned above (reduction of redundant work), and the other one of always having an up-to-date description of the item of interest. It is the duty of the project manager to see that all Analyzer entries have been made, and do exist. The above also implies that the documentor be quite familiar with the use of the Analyzer.

The individuals populating the documentation source must also have access to or a copy of the documentation schema, and must use the same section-identifiers as used in the schema. It is also feasible to have a limited amount of text formatting capability such as getting to the top of the next page, or skipping to next half page, etc. This capability would be used where a diagram, or table (which could not be drawn by the

---

[1] Part IV, "User Requirements Analyzer Command Descriptions" and Part III, "URA Outputs" for an explanation of these commands and corresponding outputs.

documentation generator) had to be inserted, and room had to be left for it. An entry line for the documentation data base is a maximum of 120 characters.

To distinguish between the various kinds of entries in the documentation source, Analyzer command entries are preceded by a "%" in column 1, text-formatting entries are preceded by a "*" in column 1, and the actual text itself has nothing preceding it, and begins in column 1. Preceding blanks are considered part of the text. An entry line for the documentation data base is a maximum of 120 characters. Text entries have no specific format.

The documentation source may be populated by the documentors in much the same way as the documentation schema (Figure 68), i.e., any processor such as the text editor, file populator, etc., may be used to store the source in the documentation data base which must be a file. If several documentors are working simultaneously, different files may be assigned to individual documentors. These portions of the data base can then be merged at documentation generation time. The order of the section entries is unimportant.

2.2 Syntax for the Documentation Source

SECTION-ENTRIES have the following syntax:

#SECTION-IDENTIFIER [HEADING-COMMENT]

The section entries are the same as in the documentation schema. However, the text portion (heading-comment) in the documentation source is ignored. Only the section-identifier is used to match against the documentation schema. The "#" must be in column 1.

ANALYZER-COMMAND-ENTRIES have the following syntax:

%ANALYZER-COMMAND

Analyzer commands may be interspersed anywhere in the text with the provision that they be preceded by a "%" in column 1. Any Analyzer commands from Part IV[1] are allowed except the STOP command and the SET command with the option PROMPT=ON in effect (by default it is not in effect). The commands are not executed immediately. They are just stored as is until the documentation generation process.

TEXT-FORMATTING-ENTRIES have the following syntax:

*CONTROL-INFORMATION-ENTRY

---

[1] Part IV, "User Requirements Analyzer Command Description."

The text formatting commands available to use in the schema will also be available to use in the source. These commands were explained in Section 1.2. They may be interspersed anywhere in the documentation source. The "*" must be in column 1.

COMMENT-ENTIRES have the following syntax:

&COMMENT

Comments may be interspersed anywhere in the documentation source. They will not appear in the final output (same as in the schema). The "&" must be in column 1.

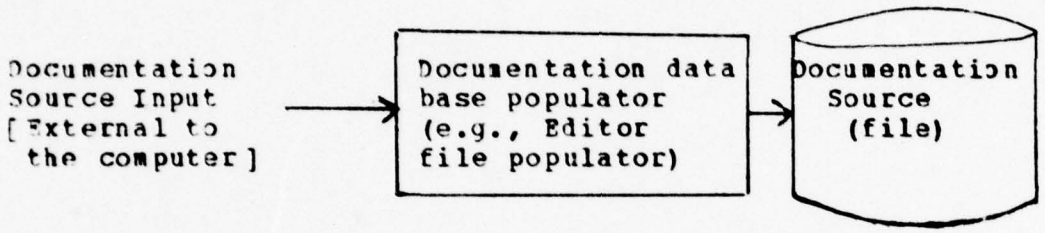

Documentation
Source Input
[External to
the computer]
Documentation data
base populator
(e.g., Editor
file populator)
Documentation
Source
(file)

Figure 68

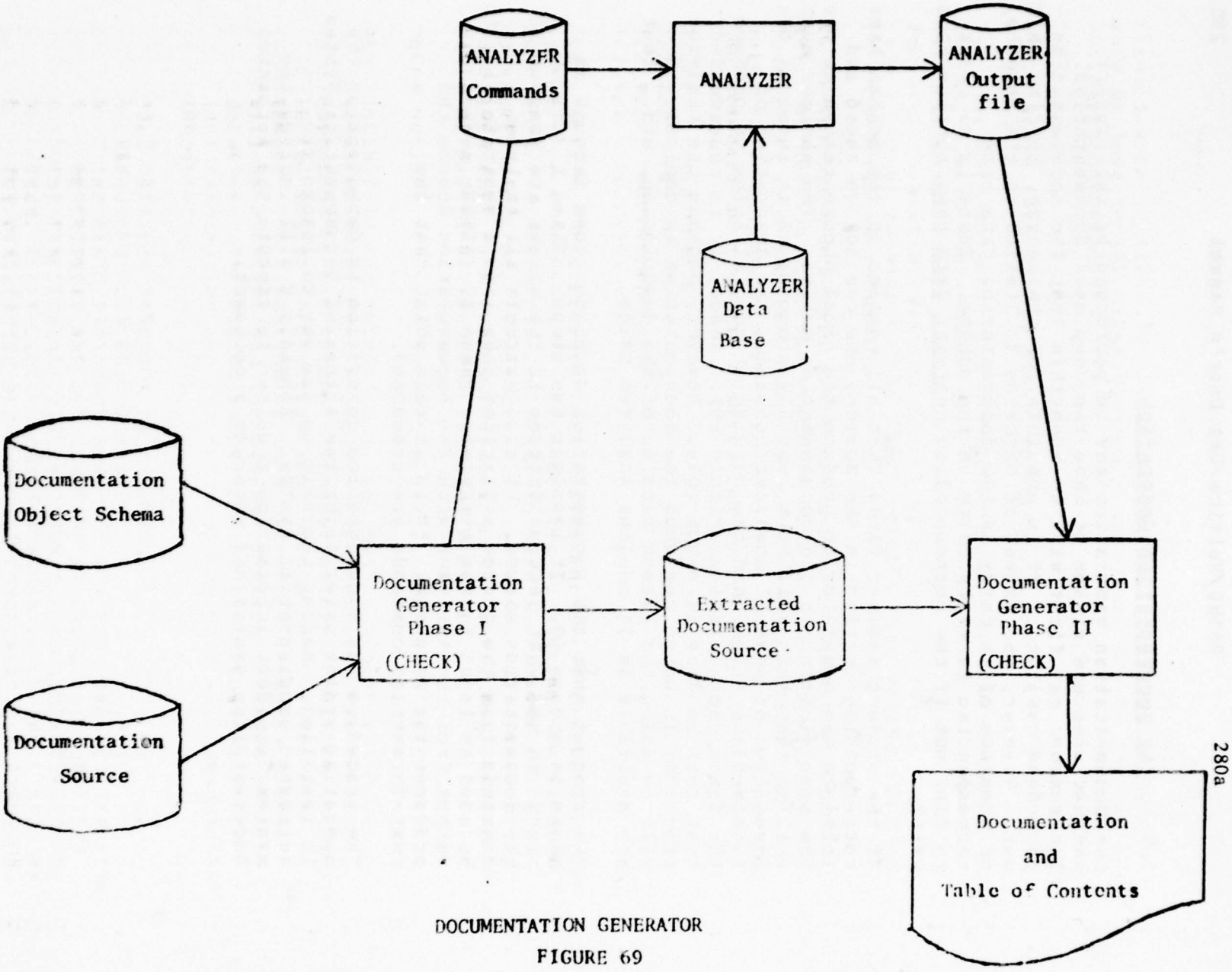
3.0 The Documentation Generation

The documentation generation may be performed by the project manager when one wishes to have the completed documentation. The manager may first wish to ascertain that the documentation is indeed complete. It is possible for the manager to find this out. However, the extent of checking is limited to the absence, or presence of an entry in the documentation data base corresponding to every entry in the schema. There is no check to find out if the corresponding Analyzer data base entries also exist.

If the project manager finds that all entries in the schema have corresponding entries in the source, one can now go ahead and initiate the generator to produce the final documentation as per the specification in the documentation schema. The manager must realize that this will be a lengthly process, as it involves the invocation of several analyzer commands, the generation of intermediate files, and the merging of several files to produce the final documentation (Figure 69). Hence, it is recommended that this be done in batch mode. However, because the Analyzer processor is being run when the documentation is desired, it will be using the latest version of the description, etc., that are available as in on-line Analyzer usage.

The process that the Documentation Generator goes through is shown in Figure 69. It involves two steps. Phase I (CHECK) checks to see what Section Entries in the schema are present in the documentation source. It also extracts all Analyzer commands from the source and writes them into a separate file to be used as input to the Analyzer. Phase II (MERGE) merges the output from the Analyzer with the documentation source and produces the document. It is at this point that the text-formatting commands are processed.

The procedure for documentation generation is dependent on the operating system under which the Automated Documentation System is installed. Hence, there is no one way to invoke the Automated Documentation System. Appendix H will include the system dependent information required to execute the Automated Documentation System and generate a document.



DOCUMENTATION GENERATOR

FIGURE 69

4.0 Usage of the Documentation Generator

In using the documentation generator, there are several areas that need to be considered in order to achieve a desired quality level. Coordination between the schema, Documentation data base and Analyzer data base are mandatory in order to develop a document which will contain complete information. Development of the methods to produced these three inputs into the system so that they can be used in an ongoing operation (camera-ready documentation) is the topic of this section. Considerations as to what should be included in each component, what pitfalls should be watched out for, and what variations might arise in usage of the documentation generator will be discussed. It is important for the reader to realize at this point that while this system has been developed to provide a generalized format for producing documents under a set of standards, the system still allows a great deal of flexibility to the user.

Department of Defense Manual 4120.17-M along with MIL standard 483 will be used as example documentation standards in this discussion. Development of a Functional Description and Data Requirements Document of the Pay System Analyzer data base (W.P. 74)[1] will be used as a specific example (Appendix F).

4.1 Development of the Documentation Schema

The documentation schema consists of a structural outline or table of contents of the particular document being produced. From this schema a table of contents for the document being generated will be produced. Hence, it is important that the user include enough section levels in order to make the table of contents meaningful. It is also important to note that the final documentation will be produced in the order stated in the documentation schema, so the schema should be set up appropriately.

Format for the schema is discussed in Section 1. The documentation initializer (see Figure 64) produces the documentation and schema which is stored permanently on file after any required editing is done.

Within the schema, comments and formatting commands are allowed. The schema is checked to see if it contains any formatting commands. If it does, each formatting command goes into effect prior to the printing of the section header and immediately following the formatting command. This is also true for commands at the beginning of the schema.

Although it is suggested that the table of contents of a particular document or standard be used as the schema, many

---

[1] "An Example of the Use of PSL Using Top-Down Analysis."

times a more detailed schema is warranted. Situations where more or less detail might be deemed necessary are in the following subsections.

4.1.1 Where Less Detail Might Be Necessary

The table of contents of a certain standard is a skeleton type set up, giving suggestions on how the documents should be developed, but leaving room for variances in structure from one document to another written according to the standard. In this case it might be wise to use less detail in the schema and keep it as general as possible.

An example of where this situation arises is in MIL standard 483. In Section 3.2, the standard sets up the criteria for describing each separate function of a system. Since there is going to be a varying number of functions in different systems, the standard sets up the description as follows:

```
3.2.X      Function X
3.2.X.1    INPUTS
3.2.X.2    PROCESSING
3.2.X.3    OUTPUTS
```

Thus, this basic structure is repeated for every function in the system. This causes a definite problem when using the document generator. Since it is desirable to have one schema for all documents produced, it can be seen that since the number of functions will vary from one system to the next, the standard's table of contents cannot be followed strictly.

Three options are open to the documentor in cases like this.

a) Include in the schema only the section header (#3.2 DETAILED FUNCTIONAL REQUIREMENTS) and leave the rest to be described in the source. Comment entries could be included in the schema to describe what should be included with these. The advantage of this option is that the schema will be constant and independent of each instance of this document. The disadvantage is that the schema will now lack information and the table of contents will be somewhat incomplete.

```
#3.2 DETAILED FUNCTION REQUIREMENTS
&  PARAGRAPH 3.2X FUNCTION X.
&    THE BASIC PARAGRAPH FOR EACH FUNCTION SHALL BEGIN WITH
&    DESCRIPTIVE AND INTRODUCTORY MATERIAL WHICH DEFINES
&    THE FUNCTION AND ITS RELATIONSHIP TO OTHER FUNCTIONS.
&    THEN, THE FOLLOWING THREE SUBPARAGRAPHS SHALL SPECIFY
&    THE QUALITATIVE REQUIREMENTS CONCERNING THE FUNCTION.
&  PARAGRAPH 3.2.X.1  INPUTS.
&  PARAGRAPH 3.2.X.2  PROCESSING.
&  PARAGRAPH 3.2.X.3  OUTPUTS.
```

b) Set up the schema for an arbitrary number of functions. An example of this would be:

```
#3.2.1     FUNCTION ONE
#3.2.1.1   INPUTS
#3.2.1.2   PROCESSING
#3.2.1.3   OUTPUTS
#3.2.2     FUNCTION ONE
#3.2.2.1   INPUTS
#3.2.2.2   PROCESSING
#3.2.2.3   OUTPUTS
      .
      .
      .
#3.2.12    FUNCTION TWELVE
#3.2.12.1  INPUTS
#3.2.12.2  PROCESSING
#3.2.12.3  OUTPUTS
```

The advantage of this option is that the table of contents is complete. The disadvantage is that the entries in the table are not meaningful.

c) Include in the schema the actual function names for each system. An example of this would be:

```
#3.2       DETAILED FUNCTION REQUIREMENTS
#3.2.1     SENSOR CALIBRATION (INITIAL) FUNCTION
#3.2.1.1   INPUTS
#3.2.1.2   PROCESSING
#3.2.1.3   OUTPUTS
#3.2.2     ELEMENT CORRECTION (INITIAL) FUNCTION
#3.2.2.1   INPUTS
#3.2.2.2   PROCESSING
#3.2.2.3   OUTPUTS
#3.2.3     MANEUVER DETERMINATION CONTROL FUNCTION
#3.2.3.1   INPUTS
#3.2.3.2   PROCESSING
#3.2.3.3   OUTPUTS
#3.2.4     SIMULTANEOUS SOLUTION OF MANEUVER AND ELEMENT
          FUNCTIONS
#3.2.4.1   INPUTS
#3.2.4.2   PROCESSING
#3.2.4.3   OUTPUTS
#3.2.5     ELEMENT MAINTENANCE CONTROL FUNCTION
#3.2.5.1   INPUTS
#3.2.5.2   PROCESSING
#3.2.5.3   OUTPUTS
#3.2.6     ELEMENT CORRECTION (ROUTINE) FUNCTION
#3.2.6.1   INPUTS
#3.2.6.2   PROCESSING
#3.2.6.3   OUTPUTS
#3.2.7     OBSERVATION EDITING FUNCTION
```

#3.2.7.1 INPUTS
#3.2.7.2 PROCESSING
#3.2.7.3 OUTPUTS
#3.2.8 SENSOR CALIBRATION (ROUTINE) FUNCTION
#3.2.8.1 INPUTS
#3.2.8.2 PROCESSING
#3.2.8.3 OUTPUTS
#3.2.9 ELEMENT RECOVERY CONTROL FUNCTION
#3.2.9.1 INPUTS
#3.2.9.2 PROCESSING
#3.2.9.3 OUTPUTS
#3.2.10 ELEMENT CORRECTION (RECOVERY) FUNCTION
#3.2.10.1 INPUTS
#3.2.10.2 PROCESSING
#3.2.10.3 OUTPUTS
#3.2.11 MANEUVER DETECTION FUNCTION
#3.2.11.1 INPUTS
#3.2.11.2 PROCESSING
#3.2.11.3 OUTPUTS
#3.2.12 MANUAL INTERACTIVE DIFERRENTIAL CORRECTION FUNCTION
#3.2.12.1 INPUTS
#3.2.12.2 PROCESSING
#3.2.12.3 OUTPUTS

The advantage of this method is that the schema is complete and descriptive. The disadvantage is that the schema will have to be changed for each application document.

The decision as to which one of these options should be used is up to the discretion of the user.

4.1.2 Where More Detail Might Be Necessary

If within the standard, references are made to information that is deemed necessary to include within a certain section, it might be advisable to include this information in the schema. An example of this would be in the Data Requirements Document of D.O.D. Manual 4120-17-M. In this document, the table of contents looks as follows:

1. GENERAL DATA REQUIREMENTS
1.1 PURPOSE OF DATA REQUIREMENTS
1.2 PROJECT REFERENCES
1.3 MODIFICATION OF DATA REQUIREMENTS
2. DATA DESCRIPTION
2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
2.2 LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
2.4 INTERNALLY GENERATED DATA
2.5 SYSTEM DATA CONSTRAINTS
3. USER SUPPORT FOR DATA COLLECTION
3.1 DATA COLLECTION REQUIREMENTS AND SCOPE

3.2 RECOMMEND SOURCE OF INPUT DATA
3.3 DATA COLLECTION AND TRANSFER PROCEDURES
3.4 DATA BASE IMPACTS

Within Section 3.1, the standard refers to nine areas of supplementary information. If this information is considered to be of such a nature that it will or should be included in most all the documents being produced, the user might want to include the information in the schema (therefore in the table of contents of the document). The schema would then look like:

```
#1.       GENERAL DATA REQUIREMENTS
#1.1      PURPOSE OF DATA REQUIREMENTS
#1.2      PROJECT REFERENCES
#1.3      MODIFICATION OF DATA REQUIREMENTS
#2.       DATA DESCRIPTION
#2.1      LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
#2.2      LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
#2.3      LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
#2.4      INTERNALLY GENERATED DATA
#2.5      SYSTEM DATA CONSTRAINTS
#3.       USER SUPPORT FOR DATA COLLECTIONS
#3.1      DATA COLLECTION REQUIREMENTS AND SCOPE
#3.1.1    INPUT SOURCE(S) OF THE DATA ELEMENT
#3.1.2    INPUT MEDIUM
#3.1.3    RECIPIENTS
#3.1.4    CRITICAL VALUE
#3.1.5    SCALES OF MEASUREMENT
#3.1.6    CONVERSION FACTORS
#3.1.7    OUTPUT FORM/DEVICE
#3.1.8    EXPANSION FACTORS
#3.1.9    FREQUENCY OF UPDATE
#3.2      RECOMMEND SOURCE OF INPUT DATA
#3.3      DATA COLLECTION AND TRANSFER PROCEDURES
#3.4      DATA BASE IMPACTS
```

It might be that the documentor just needs to be reminded of the supplementary information. In this case it might be appropriate just to include an appropriate comment.

```
#1.    GENERAL DATA REQUIREMENTS
#1.1   PURPOSE OF DATA REQUIREMENTS
#1.2   PROJECT REFERENCES
#1.3   MODIFICATION OF DATA REQUIREMENTS
#2     DATA DESCRIPTION
#2.1   LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
#2.2   LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
#2.3   LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
#2.4   INTERNALLY GENERATED DATA
#2.5   SYSTEM DATA CONSTRAINTS
#3.    USER SUPPORT FOR DATA COLLECTION
#3.1   DATA COLLECTION REQUIREMENTS AND SCOPE
&      AT THIS POINT IT SHOULD BE NOTICED THAT THERE ARE NINE
       AREAS TO BE CONCERNED WITH IN THIS SECTION.  THEY ARE:
```

```
&       A.  INPUT SOURCE(S) OF THE DATA ELEMENT
&       B.  INPUT MEDIUM
&       C.  RECIPIENTS
&       D.  CRITICAL VALUE
&       E.  SCALES OF MEASUREMENT
&       F.  CONVERSION FACTORS
&       G.  OUTPUT FORM/DEVICE
&       H.  EXPANSION FACTORS
&       I.  FREQUENCY OF UPDATE
#3.2  RECOMMEND SOURCE OF INPUT DATA
#3.3  DATA COLLECTION AND TRANSFER PROCEDURES
#3.4  DATA BASE IMPACTS
```

The decision as to what should be in the schema is based primarily on how much detail is wanted and/or expected in the table of contents.

4.2 Development of the Documentation Source

The format of the documentation source input and a description of how it is transformed into the documentation source is in Section 2. There are several areas that the user need be aware of when developing the documentation source. Proficiency in these areas is necessary if the documentor is to attain full range capabilities of the document generator.

4.2.1 Knowledge of the Analyzer Data Base

It is of utmost importance that the documentor have a thorough knowledge of the Analyzer data base as well as the Analyzer command language. The documentor must be able to pick out the information needed for the document. When it is known that a system will have to be documented using a certain standard, it is also important that the person developing the Analyzer data base have a full understanding of what information is needed to fulfill the standard's documentation requirements. Since the standard usually requires complete information about the system, writing the Analyzer data base with the standard in mind will usually serve to insure a complete system description.

4.2.2 Text Material

The documentation source for various projects written under a certain standard should be developed in such a way as to minimize the differences in the various Documentation sources. This can cut down substantially on the documentor's time spent on each system. Another advantage of doing this is that once a certain combination of commands has been determined as the best way to convey certain information required by a particular standard, then that combination can be repetitively used for all projects documented under this particular standard.

With this in mind, the documentor should try to include in the source only such text that will be present in every instance of the document being produced. Examples of such text are in paragraphs found in both the Functional Description and Data Requirements Documents. These paragraphs describe the purpose of each of the documents:

Section 1.1

> This Functional Description for (Project Name) (Project Number) is written to provide:
>
> a) The system requirements to be satisfied which will serve as a basis for mutual understanding between the user and the developer.
>
> b) Information on performance requirements, preliminary design, and user impacts, including fixed and continuing costs.
>
> c) A basis for the development of system tests.

Section 1.2

> The objectives of this Data Requirements Document for (Project Name) (Project Number) are to list and define data elements which the system must handle and to communicate data collection requirements to the user.

Another examples of where text should be used is when a common portion of the document is being described. An example of this is Section 3.3 of the Functional Description. Here is an example of how the source might look:

```
#3.3 INPUTS/OUTPUTS
     INPUTS:
%NG  S='INPUT'
%FPS
*SKIP 2
     OUTPUTS:
%NG  S='OUTPUT OR INPUT'
%FPS
%STR OUTPUT
```

Since the documentation source data base will be different for each project documented, it is not necessary that the text be constant. However, this practice reduces redundancy in work done by the documentor.

4.2.3 Analyzer Commands

As mentioned before, the documentor must have a full understanding of the Analyzer command capabilities (described in Part IV[1]) including all possible options. Thus knowledge is essential for the document generator to produce complete information in a readible form. Some of the commands and options that might be useful are listed below:

PCOM	Options:	DESCRIPTION NOPUNCH
NG	Options:	SUBLEVEL=# SUBPART-OF=NAME NOPRINT,PRINT KEY=keyword-name
CONTENTS		
PICTURE	Options:	NODATA NOFLOW NOSTRUCTURE

The Data Process Report could be very beneficial, especially since it produces a very complete description of data flow and process functions. It can be used when an overall interaction description is needed. This report also shows if consistency is present as well as showing completeness (or incompleteness).

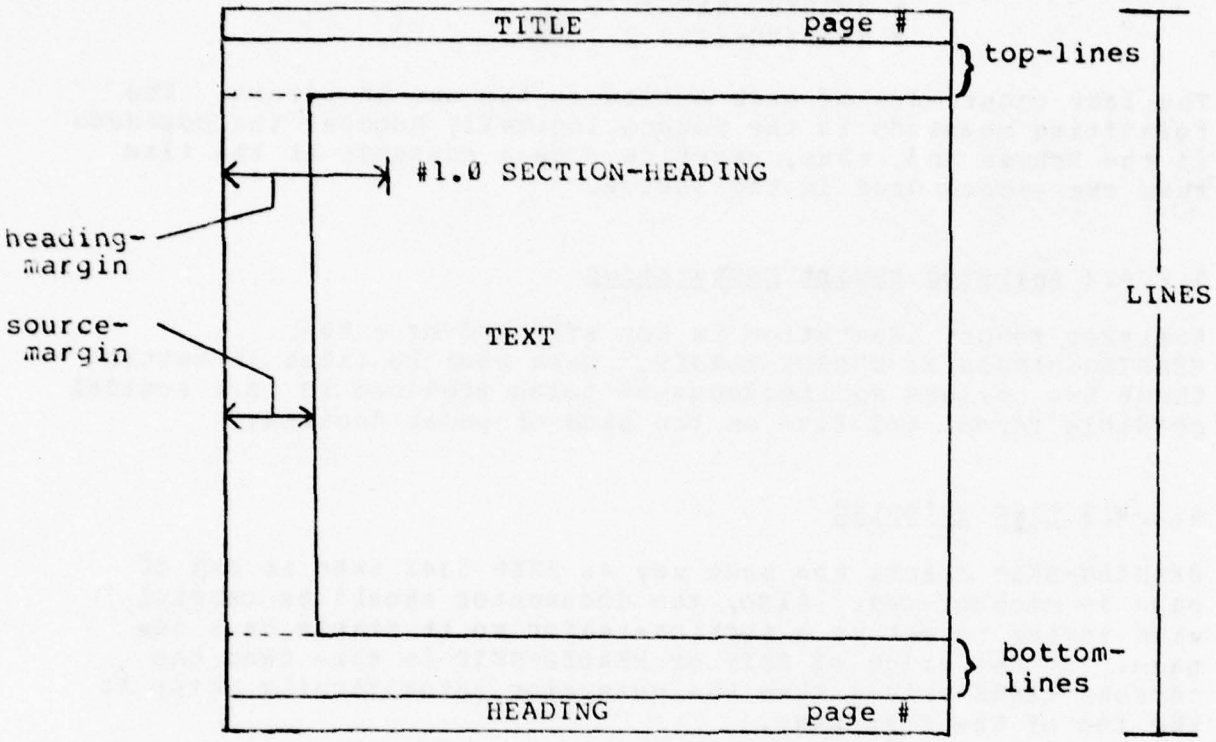
The STOP command should not be used. There are three options of the SET command that should not be used: PROMPT=ON will virtually destroy the document being produced. It is also recommended that OUTPUT=XXX, HEADING=ON, PARM=ON not be used. (All four of these options default to OFF.)

---

[1] Part IV, "User Requirements Analyzer Command Description."

4.2.4 Formatting Entries

As shown in Section 2.2, the text formatting entries allow the user a great deal of maneuverability in regards to the final document format. At this point, a diagram might help to show just what is going on a page of a document.



It is important for the documentor to understand the formatting commands. There are several points that should be remembered when using the commands. These are covered in the following subsection.

4.2.4.1 Defaults of Formatting Commands

The formatting options can be changed at any time in the Source. The formatting commands defaults affect how the document will appear. If one does not specify a value for these commands, then the default will be used. When the documentor does not specify these command values, then the final document will appear as if the defaults were included in the Source data base. The formatting command defaults are:

```
* REPORT-CC ON
* TITLE OFF
* HEADING ON
* TOP-LINES 3
* BOTTOM-LINES 2
* LINES 60
* SOURCE-MARGIN 5
* HEADING-MARGIN 5
* HEADING-SKIP 2
```

The last occurrence of each option is the one in effect. The formatting commands in the Source logically succeed the commands in the Schema and, thus, override Schema commands at the time they are encountered in the Source.

4.2.4.2 Analyzer Report Indentation

Analyzer report identation is not affected by either HEADING-MARGIN or SOURCE-MARGIN. Care must be taken in setting these two options so the document being produced is in a logical readible format and fits on the size of paper desired.

4.2.4.3 LINE SKIPPING

HEADING-SKIP reacts the same way as SKIP does when an end of page is encountered. Also, the documentor should be careful when trying to set up a section-header so it starts on a new page. If the value of SKIP or HEADER-SKIP is more than the current LINES values then the generator automatically skips to the top of the next page.

4.2.4.4 Usage of REPORT-CC

The REPORT-CC command specifies whether to use the Analyzer REPORT Carriage Control symbols or to suppress them. REPORT-CC ON means that the normal carriage control for the Analyzer reports will be used. OFF means that they will be suppressed and the usual paging sequence will be used.

Usage of the REPORT-CC command will have a direct bearing on the paging of the document being processed. Since when REPORT-CC is ON, analyzer reports will be produced according to their normal paging sequence. This is helpful when a PICTURE Report is being produced and more than one name is input or the PICTURE must be continued on the next page. Other reports where this option is helpful are the DATA-PROCESS Report and the CONSISTS-COMPARISON Report. In cases where a report will be taking up more than one page, it is important to start the report at the top of a fresh page so that the Analyzer paging will match the documentation generator's paging. There are cases, though, where the documentor will wish to set REPORT-CC OFF, such as where several

names are being entered into the FPS Report or CONTENTS Report.

4.2.4.5 The HOLD and HOLD-BLANK Commands

The documentor should understand the usefulness of the HOLD and HOLD-BLANK commands. If a diagram, table or figure must be drawn manually, the HOLD-BLANK command should always be used. This will insure enough space is saved on one complete page to insert the figure. If an analyzer report or some text material is going to be produced and the documentor wishes that this material be contained on one page, the HOLD command should be used. An example of where it would be used is with the PICTURE Report. Since the PICTURE Report uses 41 lines, the documentor should place a HOLD 41 preceding the PICTURE command to insure that it is produced on a single page.

4.2.5 Usage of Analyzer SYNONYMS

A SYNONYM is a short abbreviation form of a name that can be stored in the Analyzer data base. A SYNONYM is a reserved word of the Analyzer.

In order to generalize the documentation Source data base, SYNONYMS should be used whenever possible to describe certain items, such as main functions, master files, etc. The implication of this practice on the Analyzer data base formulation is discussed in section 4.3.2.

By using SYNONYMS, the documentor is relieved of changing even more of the Source data base from one project to the next. An example of how this can be done is in section 3.2 of the Functional Description. The Source data base will look as follows:

```
#3.2 SYSTEM FUNCTIONS
%NG  S='MAIN-PROCESS AND SL=1'
       THE VARIOUS SUB-FUNCTIONS OF THE SYSTEM ARE:
%PCOM DESC PRCD NOPUNCH
```

In this example MAIN-PROCESS is a SYNONYM.

Section 2.4.2.4 is another example.

```
#2.4.2.4 OPERATIONAL IMPACTS
     A.  OPERATIONAL STRUCTURE
%STR
*SKIP 2
     B.  TIMELINESS(OPERATIONS AND DATA)
%FREQ
*SKIP 2
     C.  INPUTS
%STR INPUT
```

```
*SKIP 2
      D.  DATA RETENTION
          1.  MASTER FILE
%PCOM N= MASTER-FILE DESC DER NOPUNCH
*NEW-PAGE
%PIC N= MASTER-FILE NODATA NOFLOW
```

Here, MASTER-FILE is a SYNONYM.

4.3 Development of the Analyzer Data Base with the Document Generator in Mind[1]

There are two situations that can occur in relation to the Analyzer data base when using the Document Generator.

The Analyzer data base has either been written with the document to be produced in mind, or it has not been. In any case, it is going to be beneficial for the documentor to write or update the Analyzer data base in such a way as to facilitate ease of production of the document using the generator. Various practices can be used by the documentor to accomplish this. These practices are explained in the following subsections.

4.3.1 Usage of MEMO

When it is necessary to describe certain aspects of a propsed system that are general in nature but do not apply directly to the structural, functional, or data flow of the system, then the usage of MEMOs in the Analyzer data base can be very helpful. When objectives, background material, requirements, impacts, etc., are needed, a MEMO is the obvious answer. Hence, simple PUNCH-COMMENT-ENTRY is all that is needed in the documentation source. An example of a typical MEMO is shown below. This MEMO relates the objectives required for Section 2.2 of the Functional Description:

```
MEMO                                          OBJECTIVES-MEMO;
   DESCRIPTION;
     THIS PROJECT HAS BEEN AUTHORIZED TO DEVELOP A PAY SYSTEM
     FOR THIS ORGANIZATION.  THIS PAY SYSTEM WILL PERFORM
     PAYROLL PROCESSING USING EMPLOYEE INFORMATION COMING FROM
     DEPARTMENTS AND EMPLOYEES AND WILL PRODUCE OUTPUTS WHICH
     WILL GO TO THE DEPARTMENTS AND EMPLOYEES.  THE SYSTEM WILL
     ALSO MAINTAIN THE PAYROLL MASTER INFORMATION.
```

4.3.2 Usage of NAMES and SYNONYMS

---

[1] Note: All capitalized names in this section refer to URL Reserved Words. For a more detailed explanation, refer to Parts I and II of the URL User's Manual.

When the documentor or analyst is constructing Analyzer data bases, one should keep in mind what Analyzer NAMES and SYNONYMS have been used in previous documentation source. The documentor's work can be reduced if common names are used to describe similar sections of Analyzer data bases which are describing systems that will have to be documented under the same standard. For instance, a documentor might want to include sections called BACKGROUND-MEMO and OBJECTIVE-MEMO when developing Analyzer data bases to be documented using the Functional Description Standards. The documentor might also want to have an ATTRIBUTE named DEVELOPMENT-TIMES in these Data Bases. An example of ATTRIBUTE usage follows:

```
DEFINE                                  DEVELOPMENT-TIMES
     AS A ATTRIBUTE;
/* VALUES ARE:
        MAN-HOURS-120
   FOR                          PROGRAMMING,
        PLENTY FOR              COMPUTER-TIME,
        MAN-HOURS-100
   FOR                          DATA-BASE-DEVELOPMENT,
        MAN-HOURS-20
   FOR                          PAYROLL-CLERICAL,
        MAN-HOURS-30
   FOR                          OPERATIONS,
   */
```

SYNONYMS can be used in similar fashion. If the documentor uses similar synonyms in each data base, his editing can be reduced. Examples of this are using MASTER-FILE as a SYNONYM for the main file of the system and using MAIN-PROCESS as a synonym for the highest level function in the system.

4.3.3 Completeness

There are three areas in which the documentor or analyst should strive to achieve completeness in the Analyzer data base. These areas are data flow, system structure, and functional flow.

Data flow should be complete in that all files or data sets are accounted for. This means that data definition should be included. Elements of data sets need be defined only to the level of description required. The CONTENTS Report is a good means of checking data description completeness. Usage of UPDATES, DERIVES, GENERATES, RECEIVES, and USES statements is the vehicle by which data flow information is presented.

System structure must be complete for obvious reasons. The STRUCTURE Report is helpful in checking and representing the information.

Functional flow is perhaps the trickiest facet in the

development of the Analyzer data base. Care must be taken to insure that a functional chain of EVENTS, CONDITIONS and PROCESSES exists in a meaingful fashion. Usage of TRIGGERS, ON TERMINATION OF, ON INCEPTION OF, TRIGGERS, HAPPENS, and WHEN statements is suggested.[1]

4.3.4 DESCRIPTIONS

The documentors should keep in mind that in many cases standard will need descriptions in the form of text on different aspects of systems. The DESCRIPTION statement can be used by the documentor to meet these text requirements. The documentor should include a DESCRIPTION with all major sections of an Analyzer data base, as well as any other descriptive statements that are deemed necessary (i.e., PROCEDURE, VOLATILITY-SET, etc.).

4.3.5 Usage of KEYWORDS

When one or more objects in a system description need to be identified for selection and analysis purposes, KEYWORDS should be attached to each of the objects. By doing this, the documentor has a link between related objects. For example, if all functions (PROCESSES) described as being manual procedures in a system description were to be listed and analyzed together, the KEYWORD "manual" could be attached to each PROCESS for this purpose. By doing a NAME-GEN with the option KEY=MANUAL, a list of these PROCESSES could then be produced.

---

[1] See Part I for an explanation of these terms.

Appendix A

URA COMMAND ABBREVIATIONS

This appendix presents all URA commands, with their parameters and defaults. It also presents the acceptable abbreviations for these commands and parameters. There are several conventions in choosing these abbreviations that may aid the user:

1. For command names that consist of only one word, for example, CONTENTS or PICTURE, the first three or four letters of the name are used as their abbreviation. CONT is the abbreviation for CONTENTS and PIC is the abbreviation for PICTURE.

2. For most command names that consist of more than one word, e.g., CONSISTS-MATRIX or FORMATTED-PROBLEM-STATEMENT, the abbreviation is derived by using the first letter from each word in the command name. This gives CM for CONSISTS-MATRIX and FPS for FORMATTED-PROBLEM-STATEMENT. This convention is not strictly followed however, so that the abbreviations may be more meaningful. For example, DCOM is the abbreviation for DELETE-COMMENT-ENTRY which is more mnemonic than DCE.

3. Abbreviations for parameters used in the same way by several commands are the same. The FILE parameter then, always has an abbreviation, F, allowed no matter which command is using it. Some of the more common parameters are listed below:

Parameter	Abbreviation
FILE	F
NAME	N
INPUT	I
NOPRINT	NP
PUNCH	P

4. Whenever an abbreviation exists for a parameter that has a "NO" prefix, such as NOPRINT, the abbreviation is always prefixed with "N". For example, NP is the abbreviation for NOPRINT.

A blank entry in the "Parameters" column for a command designates that there are no parameters for the command. A blank entry in the "Abbreviations" column designates that there is no abbreviation for the command or parameter name. A blank entry in the "Defaults" column designates that there is no default for this parameter for the given command.

Command Name	Parameters	Abbreviations	Defaults
CHANGE-TYPE		CT	
	FILE	F	
	NAME	N	
	TYPE	T	
CONSISTS-COMPARISON		CNC	
	FILE	F	FILE
CONSISTS-MATRIX		CM	
	FILE	F	FILE
	NAME	N	
	CONTAINED	CNTD	
	CONSISTS	CSTS	
CONTENTS		CONT	
	FILE	F	FILE
	NAME	N	
	INDEX		
	NOINDEX		NOINDEX
	LEVELS		LEVELS=ALL
	NCFLAG		
	NONCFLAG		NONCFLAG
DATA-PROCESS		DP	
	FILE	F	FILE
	NAME	N	
	DATA	D	
	PROCESS	P	
	DPMAT		DPMAT
	NODPMAT		
	DPANL		DPANL
	NODPANL		
	PMAT		PMAT
	NOPMAT		
	PANL		PANL
	NOPANL		
DELETE		DEL	
	FILE	F	
	NAME	N	
DELETE-COMMENT-ENTRY		DCOM	
	DESCRIPTION	DESC	
	NODESCRIPTION	NDESC	NODESCRIPTION
	DERIVATION	DER	
	NODERIVATION	NDER	NODERIVATION
	FALSE-WHILE	FW	
	NOFALSE-WHILE	NFW	NOFALSE-WHILE
	PROCEDURE	PRCD	
	NOPROCEDURE	NPRCD	NOPROCEDURE
	TRUE-WHILE	TW	

	NOTRUE-WHILE	NTW	NOTRUE-WHILE
	VOLATILITY	VOL	
	NOVOLATILITY	NVOL	NOVOLATILITY
	VOLATILITY-MEMBER	VOLM	
	NOVOLATILITY-MEMBER	NVOLM	NOVOLATILITY-MEMBER
	VOLATILITY-SET	VOLS	
	NOVOLATILITY-SET	NVOLS	NOVOLATILITY-SET
	FILE	F	
	NAME	N	
	PRINT		PRINT
	NOPRINT	NP	
DELETE-PSL		DPSL	
	INPUT	I	INPUT=terminal
	SOURCE	S	SOURCE
	NOSOURCE	NS	
	XREF	X	
	NOXREF	NX	NOXREF
DICTIONARY		DICT	
	FILE	F	FILE
	NAME	N	
	INDEX		
	NOINDEX		NOINDEX
	NUM-SPACE	NS	NUM-SPACE=2
	DESCRIPTION	DESC	DESCRIPTION
	NODESCRIPTION	NDESC	
	KEYWORDS	KEY	KEYWORDS
	NOKEYWORDS	NKEY	
	RESPONSIBLE-PD	RPD	RESPONSIBLE-PD
	NORESPONSIBLE-PD	NRPD	
	SYNONYMS	SYN	SYNONYMS
	NOSYNONYMS	NSYN	
ENTITY-IDENTIFIER		EI	
	FILE	F	FILE
	NAME	N	
	IDENTIFIER	I	
	ENTITY	E	
EXTENDED-PICTURE		EP	
	FILE	F	FILE
	NAME	N	
	INDEX		
	NOINDEX		NOINDEX
	STRUCTURE	STR	
	DATA-FLOW	DF	
	FORWARD	FWD	
	BACKWARD	BWD	
	DOWNWARD	DOWN	
	UPWARD	UP	
	LINKS		LINKS=1000

	COLUMNS	COLS	COLUMNS=119
	ROWS		ROWS=39
	HORIZONTAL-BOXES	HB	
	VERTICAL-BOXES	VB	
FORMATTED-PROBLEM-STATEMENT		FPS	
	ALL-STATEMENTS	AS	AS
	NOALL-STATEMENTS	NAS	
	AMARG	AM	AMARG=10
	BMARG	BM	BMARG=25
	COMMENT	COM	COMMENT
	NOCOMMENT	NCOM	
	COMPLEMENTARY-STATEMENTS	COMP	COMP
	NOCOMPLEMENTARY-STATEMENTS	NCOMP	
	CMARG	CM	CMARG=1
	DEFINE	DEF	DEFINE
	NODEFINE	NDEF	
	DESG	DG	DESG
	NODESG	NDG	
	EMPTY		
	NOEMPTY		
	FILE	F	FILE
	NAME	N	
	HMARG	HM	HMARG=40
	INDEX		
	NOINDEX		NOINDEX
	LINE-NUMBERS	LNS	LNS
	NOLINE-NUMBERS	NLNS	
	NEW-LINES	NL	
	NONEW-LINES	NNL	NONEW-LINES
	NEW-PAGE	NPG	
	NONEW-PAGE	NNPG	NONEW-PAGE
	NMARG	NM	NMARG=20
	ONE-PER-LINE	OPL	ONE-PER-LINE
	SEVERAL-PER-LINE	SPL	
	PRINT		PRINT
	NOPRINT	NP	
	PRINT OF	PEOF	PEOF
	NOPRINT OF	NPEOF	
	PUNCH	P	
	NOPUNCH		NOPUNCH
	RNMARG	RM	RNMARG=70
	SMARG	SM	SMARG=5
FREQUENCY		FREQ	
HELP			
	command-name		
	SHORT		SHORT
	LONG		

INPUT-PSL		IP	
	DBREF	D	DBREF
	NODBREF	ND	
	INPUT	I	INPUT=terminal
	SOURCE	NS	SOURCE
	NOSOURCE	S	
	UPDATE	U	
	NOUPDATE	NU	NOUPDATE
	XREF	X	
	NOXREF	NX	NOXREF
KWIC			
	FILE	F	FILE
	DIF		DIF=2
NAME-GEN		NG	
	EMPTY		
	NOEMPTY		
	ORDER		ORDER=BYTYPE
	PRINT		PRINT
	NOPRINT	NP	
	PUNCH	P	PUNCH
	NOPUNCH		
	SELECTION	S	
	INPUT	I	
NAME=LIST		NL	
	ORDER		ORDER-ALPHA
PICTURE		PIC	
	DATA	D	DATA
	NODATA	ND	
	FILE	F	FILE
	NAME	N	
	FLOW		FLOW
	NOFLOW		
	INDEX		
	NOINDEX		NOINDEX
	STRUCTURE	STR	STRUCTURE
	NOSTRUCTURE	NSTR	
PRINT-ATTRIBUTE-VALUES		PAV	
	FILE	F	FILE
	NAME	N	
PROCESS-CHAIN		PC	
	FILE	F	FILE
	NAME	N	
	INDEX		
	NOINDEX		NOINDEX
	LINKS		LINKS=1000
	COLUMNS	COLS	COLUMNS=119

ROWS		ROWS=39
HORIZONTAL-BOXES	HB	
VERTICAL-BOXES	VB	
PROCESS-INPUT-OUTPUT	PRIO	
FILE	F	FILE
NAME	N	
DESCRIPTION	DESC	DESCRIPTION
NODESCRIPTION	NDESC	
PROCEDURE	PRCD	
NOPROCEDURE	NPRCD	NOPROCEDURE
INPUT	INP	INPUT
NOINPUT	NINP	
OUTPUT	OUT	OUTPUT
NOOUTPUT	NOOUT	
NEW-PAGE	NPG	
NONEW-PAGE	NNPG	NONEW-PAGE
PRINT		PRINT
NOPRINT	NP	
INDEX		
NOINDEX		NOINDEX
PUNCH-COMMENT-ENTRY	PCOM	
DERIVATION	DER	
NODERIVATION	NDER	NODERIVATION
DESCRIPTION	DESC	
NODESCRIPTION	NDESC	NODESCRIPTION
FALSE-WHILE	FW	
NOFALSE-WHILE	NFW	NOFALSE-WHILE
PROCEDURE	PRCD	
NOPROCEDURE	NPRCD	NOPROCEDURE
TRUE-WHILE	TW	
NOTRUE-WHILE	NTW	NOTRUE-WHILE
VOLATILITY	VOL	
NOVOLATILITY	NVOL	NOVOLATILITY
VOLATILITY-MEMBER	VOLM	
NOVOLATILITY-MEMBER	NVOLM	NOVOLATILITY-MEMBER
VOLATILITY-SET	VOLS	
NOVOLATILITY-SET	NVOLS	NOVOLATILITY-SET
EMPTY		
NOEMPTY		
FILE	F	FILE
NAME	N	
PRINT		PRINT
NOPRINT	NP	
PUNCH	P	
NOPUNCH		NOPUNCH
SECURITY-CONSISTENCY ANALYSIS	SECA	
RESOURCE-CONSUMPTION-ANALYSIS	RCA	

RENAME		REN	
	INPUT	I	
	OLD	O	
	NEW	N	
REPLACE-COMMENT-ENTRY		RCOM	
	INPUT	I	INPUT
	PRINT		PRINT
	NOPRINT	NP	
STRUCTURE		STR	
	INDENT	IND	INDENT=3
	INDEX		
	NOINDEX		NOINDEX
	INPUT	INP	
	INTERFACE	INTF	
	OUTPUT	OUT	
	PROCESS	PROC	PROCESS
	REAL-WORLD-ENTITY[1]	RWE	
	PROCESSOR	PRCR	
SUMMARY		SUM	

[1] REAL-WORLD-ENTITY is synonymous with INTERFACE (INTF).

Appendix B

CREATING AND INITIALIZING URA DATA BASES

Before the user can add information to a URA data base, the data base must exist and be initialized properly. Creating an initialized data base in the user's working directory is a simple one-step process. The user should type

```
ec >ml>CARA>initdb (name) (size)
```

The size parameter is optional and, if given, should be the desired size of the data base, chosen from those sizes presently available. The default size if not given is 20. The name parameter is also optional and, if given, is the name that URA will use to reference the data base. Note, a suffix of .dbf should not be given by the user. The default name used if name is not given is ura. If the size parameter is to be used, then the name must also be given.

Appendix C

UTILITIES

1) Dump

To produce a card image dump of a data base:

ec >ml>CARA>dump data-base output punch

where:

data-base is the data base to be dumped (the ".dbf" qualifier is added)

output is a printed listing of the dump

punch is the card image dump segment

2) Restore

To restore a previously dumped data base:

ec >ml>CARA>restore data-base output input

where:

data-base is an empty initialized data base (the ".dbf" qualifier is added)

output is a printed listing of the input

input is the punch segment from pdum

3) Data Base Statistics

To produce a statistics report of a data base:

ec >ml>CARA>dbs data-base output

where:

data-base is the data base for the statistics report is desired.

output is a printed listing of the statistics report

4) Specification Generator

To execute the specification generator:

ec >ml>CARA>spg schema source

where:

schema is the segment containing the schema

source is the segment containing the source

5) URA Info

To obtain a short summary of URA instructions while using a terminal:

pr >ml>CARA>ura_info

Appendix D

REPORTING PROBLEMS WITH URA

Any problems arising from the use of URA software should be reported.

Problems may be reported while using Multics interactively by issuing:

```
mail * Wizard.CARA

}     message

.
```

The message should contain:

1. your name
2. your person id
3. your project id
4. the date
5. the data base you were working with
6. the command you were using
7. a short explanation of what happened

Appendix E

USING THE URA COMMAND LANGUAGE WITH MULTICS

(Version 3.2 of URA)

APPENDIX E

For Using the URA Command Language with Multics[1]

(Version 3.2 of URA)

This appendix summarizes the specific information that a user must have in order to use the Analyzer in a Multics environment. This appendix consists of several sections:

1) URA Control Commands for Multics

   DISPLAY command
   HELP command

   The HELP command has been expanded and improved for Multics users. The revised command description is included here.

   MTS command
   SET command
   STOP command

   These commands can be used when using URA under Multics (in addition to the other commands presented in ISDOS Working Paper No. 91).

2) Data Set Naming Conventions for Multics

   2.1 Naming the Data Base File
   2.2 Naming Temporary Data Sets

   This section describes what naming conventions should be followed, and what the Analyzer does to enforce the conventions.

3) Notes on Executing URA under Multics

   3.1 Creating and Initializing URA Data Bases
   3.2 Executing the Analyzer under Multics
   3.3 Example URA session

4) Multics Default Data Set Names

This part presents the default data set names used by the URA commands and referenced by ISDOS Working Paper No. 91.

5) Multics Glossary

---

[1] For a better understanding of Multics see the Multics Programmer's Manuals. For a more detailed explanation of how URA interacts with Multics, see ISDOS Working Paper No. 108 (Version 3.1/Multics R3.1).

This section explains some of the terminology used in this appendix.

A Note on Version 3.2 of URA

There are several important changes that have been made in URA which makes Version 3.2 different than Version 2.1.

First, several internal modifications (which are transparent to the user) have been made to make the URA more efficient. Second, a few errors in Version 2.1 have been corrected so that the URA commands perform properly. Third, the availability of a select number of URA commands and parameters have been removed. Last, some new commands are available in this version, and some commands have additional parameters not available in Version 2.1.

The following is a list of all modifications that fall into the latter two classes of changes:

1) The Name Generation command has been modified to permit retrieval of all names identified by a Boolean expression containing the characters + (or), &* (and), ¬ (not), and either keywords, attributes, or attribute values.

2) The Help command provides either short or long explanations of all URA commands and stand-alone utilities.

3) Some parameters have changed for the Set command, and there is a new command Display to print the values of the Set parameters. See the appendix for the description of these commands.

4) Additional parameters are available for the FPS Command to control the content and format of the print and punched output.

Command: DISPLAY Type: control command

Purpose: To display the current settings of the global switches and parameters.

Prototype: DISLAY(DIS)[parameter]...

Parameters:

ALL Default: no default
others (see below)

When the ALL parameter is given, the values of all the global switches and parameters are printed at the terminal (in interactive mode), or on the line printer (in batch mode). The parameters printed are those given in the set command description.

When any of the following parameters are given, the current value associated with it is given. This command only displays the value of the parameters: the SET command is used to change their values. The function of each parameter is given in the SET command description.

data-base(db)
echo(e)
heading(h)
lines(l)
output(o)
parameters(parm)
problem-name(pnam)
prompt(p)

Examples:
```
DISPLAY ALL
DIS DB ECHO O
```

Command: HELP[1]

Purpose: To provide the on-line user with a list of possible commands for URA or information about the parameters for a particular URA command.

Prototype: HELP [parameter]...

Parameters:

Command-name Default: (See text)

If no command-name is given, a list of currently available URA commands and utilities is given. If a command-name is given, then the parameters for that command are presented. Abbreviations for the command-name are also acceptable.

If a utility name is given, such as DUMP or RESTORE, then the description of the Multics commands necessary to execute the utility is presented. Abbreviations are PDUM and PRES.

FILES Default: (See text)

If this parameter is given, a list of the default data set names for the Multics installation is given.

SHORT, LONG Default: SHORT

If SHORT is given, only the parameters for the given command are printed. If LONG is given, explanations of the various parameters are also printed.

Examples: HELP FPS LONG

HELP FILES SHORT

---

[1] This command has been improved and expanded for the Multics installation; hence the description is included in this appendix.

Command: MTS

Purpose: To execute a Multics command(s) and return control to the Analyzer.

Prototype: MTS Multics-command

Parameters:

Multics-command line

Only the Multics-command line is executed and then control is returnel to URA. All Multics conventions apply to this command line. Multiple commands must be separated by comas. User abbreviations are permitted. Anv Multics command is allowed.

Examples: MTS LIST *.URATEMP -BR -NHE ; PWD

MTS QEDX URAPPS.URL

Notes for running under Multics:

For any single terminal session under Multics, the MTS command has the effect of exiting from the Analyzer and issuing the Mutlics command. This differs from the STOP command as the STOP terminates the URA session, and requires the exec-com command to return to URA mode. Not only does the MTS command save money (URA is not terminated), but it also retains the parameters set in the SET command and contents of the NAME-GEN defau't file, etc. A STOP would require that these parameters be restated.

The Multics change-working-directory (cwd) command must be used with caution. The Analyzer must always be running from the working directory in which it was initially invoked. The user must not perform any exec_com command that will alter the I/O attach units.

Command: SET Type: control command

Purpose: To set various global switches and parameters.

Prototype: SET [parameter]...

Parameters:

DATA-BASE(DB)=dataset-name Default: ura

The data base is the file in which the data base information is assembled and stored. This facility allows the user to change the data base being used in the middle of an Anaylzer session. The suffix .dbf should not be given. The full relative or absolute pathname should be used. No abbreviations are recognized.

ECHO(E)= {ON Default: ECHO=OFF
{OFF

With "ECHO" set equal to ON, the commands are printed on the current output device as they are encountered. This is more desirable in batch (command is printed on line printer) than in on-line mode (the command is echoed back at the terminal).

HEADING(H)= {ON Default: HEADING=ON
{OFF

With "HEADING" set OFF, printing of headings (date, title, page number, etc.) of each report will be suppressed. Headings will be printed when the switch is set ON.

LINES(L)=integer Default: LINES=46

The number of lines printed per report page is set to the indicated number. The default number fits the output to an 8-1/2 x 11 inch page for convenient binding. "LINES" may take on any value between 10 and 500.

MODE= {BATCH(B) Default: mode=terminal
{TERMINAL(T)

This switch sets other switches particular to the mode of operation such as "prompt" and "echo".

OUTPUT(O)=dataset-name Default: term

This parameter specifies the data set into which all subsequent URA reports are written. If no OUTPUT file is specified, all output is written on the terminal data set.

PARAMETERS(PARM)= {ON Default: ON
{OFF

With "PARAMETERS" set OFF, the printing of the parameters in effect for the production of each URA report will be suppressed. Parameters will be printed when the switch is set ON.

PROBLEM-NAME(PNAM)=user-name Default: Air Force
ESD/RADC
Multics

The "PROBLEM-NAME" is the title that goes on each page of the output. It must be a URL name, that is, it must begin with a code 3 character, be no more than 30 characters in length, and be composed of code 3 and 4 characters with no intervening blanks.

PROMPT(P) = {ON Default: PROMPT=ON
{OFF

If "PROMPT" is set to ON, URA will prompt the user for the correct command or parameters when an error is encountered. With "PROMPT" set to OFF, URA will ignore invalid commands and parameters and proceed to the next command or parameter.

Examples: SET DB=WP74 PNAM=W.P.74_EXAMPLE O=PRINT.FILE

Command: STOP Type: control command

Purpose: To terminate execution of the URA software and return control to Multics.[1]

Prototype: STOP

Parameters:
None

Example: STOP

---

[1] Issuing this command returns all storage used by URA for tables, buffers, parameters, scratch files, etc. All files having the suffix .uratemp are deleted. To restart execution of URA the user must give the Multics command:

exec_com >ml>CARA>ura

If the user leaves URA mode and enters Multics without use of the "STOP" command, the user should issue the Multics "start" command to return to URA.

2. Data Set Naming Conventions for URA in Multics

Version 3.2 of the Analyzer has been improved from the human engineering perspective. Specifically, the file naming conventions have been improved. The user does not have to remember as many files as has been the case with previous Analyzer versions.

2.1 Naming the Data Base File

The user no longer must commit to memory the several files associated with each data base, the tables and index, nor the suffix on the names of each file. The data base name is the name that the user wishes to use to refer to his data base. The Analyzer will attach any necessary suffix to refer to actual Multics segments. The data base may have any valid Multics segment name.

2.2 Naming Temporary Data Sets

All temporary files maintained by the Analyzer have the suffix .uratemp attached to their name. When the Analyzer is terminated, two-component segments with .uratemp as the suffix are deleted. Some PUNCH files are considered to be temporary unless the user supplies a name with a different suffix (see section 4).

Example:

The commands:

```
NAME-GEN PUNCH=MYPUNCH S="ALL"

NAME-GEN PUNCH=MYPUNCH.URATEMP S="ALL"
```

cause names to be written in the data set MYPUNCH and to MYPUNCH.URATEMP respectively. The latter data set will be deleted at the end of the session.

3. Notes on Executing URA under Multics

3.1 Creating and Initializing URA Data Bases

Before the user can add information to a URA data base, the data base must exist and be initialized properly. Creating an initialized data base in the user's working directory is a simple one-step process. The user should type:

```
ec >ml>CARA>initdb (name) (size)
```

The size parameter is optional and, if given, should be the desired size of the data base, chosen from those sizes presently available. The default size if not given is 20. The name parameter is also optional and, if given, is the name that URA will use to reference the data base. Note, a suffix of .dbf should not be given by the user. The default name used if name is not given is "ura." If the size parameter is to be used, then the name must also be given.

3.2 Executing the Analyzer under Multics

To initialize the Analyzer and enter URA mode, the following command should be given:

```
ec >ml>CARA>ura
```

The Analyzer will respond "Enter command (and any parameters)" when it is ready to accept each command. Ignore any "No I/O switch" messages.

If the user issues an interrupt to the Analyzer, then the execution may be resumed at the point of interruption by issuing the command:

```
start
```

The user should not interrupt the Analyzer during a data base update or modification command as this could leave the data base in an inconsistent state.

If the user interrupts the Analyzer during a report command or while URA is idle and wishes to prematurely terminate the Analyzer and begin again, the Multics commands are:

```
close-file-all
  release-all
>ml>CARA>closeit
```

The user must use the SET command to reset any parameters for each Analyzer session.

Example URA sessions

login Userid	[login to Multics]
password	[user password to Multics]
ec >ml>CARA>initdb mydb	[create and initialize an empty data base]
ec >ml>CARA>ura	[enter URA mode]
set db=mydb.dbf o=temp.print	[set control information]
input-psl input=afile.url update	[update URA data base using contents of afile.url in current working directory]
name-gen s='all'	[get list of all names]
fps	[get FPS for all names]
stop	[leave URA mode]
dprint temp.print	[print output - line printer]
logout	[logout of Multics]

A subsequent session may enter more input and generate some reports.

login Userid	[login to Multics]
password	[user password to Multics]
ec >ml>CARA>ura	[enter URA mode]
set db=mydb pnam=Multics-example	[set control information]
input-psl input=somefile.url update	[update the data base using the contents of somefile.url]
name-gen s='process'	[get list of process names]
picture	[get Picture report for each process name]
name-gen s='all'	[get list of all names]
fps	[get FPS for all names]
stop	[leave URA mode]
logout	[logout of Multics]

4. Multics Default Segment Names

Table A presents a summary of Multics default segment names used in conjunction with each URA command. The "Default Input File" is the default segment used as input to the command if no other segment set has been specified (i.e., via, the "INPUT=" or "FILE=" parameter.) A dash in the entry designates that the command has no Input data parameters.

The "Default PUNCH File" is the default segment used to store PUNCH data from the command if no PUNCH segment is designated via the "PUNCH=" parameter. A dash in the entry designates that no PUNCH data can be obtained directly from the command. (PUNCH data is data that is directly acceptable as input to a URA command.)

Multics Default Segment Names

Command	Default Input File	Default PUNCH File
CHANGE-TYPE	uraname.uratemp	---
CONSISTS-COMPARISON	uraname.uratemp	---
CONSISTS-MATRIX	uraname.uratemp	---
CONTENTS	uraname.uratemp	---
DATA-PROCESS	uraname.uratemp	---
DELETE	uraname.uratemp	---
DELETE-COMMENT-ENTRY	uraname.uratemp	---
DELETE-PSL	term	---
DICTIONARY	uraname.uratemp	---
ENTITY-IDENTIFIER	uraname.uratemp	---
EXTENDED-PICTURE	uraname.uratemp	---
FORMATTED-PROBLEM-STATEMENT	uraname.uratemp	urafps.url
FREQUENCY	---	---
INPUT-PSL	term	---
KWIC	uraname.uratemp	---
NAME-GEN	---	uraname.uratemp
NAME-LIST	---	---
PICTURE	uraname.uratemp	---
PRINT-ATTRIBUTE-VALUES	uraname.uratemp	---
PROCESS-CHAIN	uraname.uratemp	---
PROCESS-INPUT-OUTPUT	uraname.uratemp	uraname.uratemp
PUNCH-COMMENT-ENTRY	uraname.uratemp	urapcom.comment
RENAME	no default	---
REPLACE-COMMENT-ENTRY	urapcom.comment	---
RESOURCE-CONSUMPTION-ANALYSIS	uraname.uratemp	---
SECURITY-ANALYSIS	uraname.uratemp	---
STRUCTURE	---	---
SUMMARY	---	---

TABLE E.1

MULTICS GLOSSARY

code 3 character	A character allowed in any position in a URL name. See Appendix B.
data set	A synonym for segment.
file	A segment or multi-segment file.
multi-segment file	A file composed of multiple page units.
pathname (absolute)	The connotation of a segment's entry-name with all superior directories leading back to the storage system root.
pathname (relative)	The pathname that uniquely names a segment relative to the working directory.
quit signal	The means by which users may interrupt Multics from processing a program or command lines.
segment	The basic unit of information within the Multics storage system. Each segment has access attributes and a name, and may contain data, programs, or be null.
URA session	An interactive session between the Multics command "ec >udd>CARA>ura" and the URA command "stop."
working directory	The directory under which the user is doing his work.

Appendix F

EXAMPLE OF DOCUMENT SCHEMA AND SOURCE

This appendix consists of two examples, a Fuctional Description and a Data Requirements Document. Each example consists of a schema and a corresponding source. The examples are presented in the following order:

Schema for the Functional Description

Source for the Functional Description

Schema for the Data Requirements

Source for the Data Requirements

```
 1 *TITLE FUNCTIONAL DESCRIPTION
 2 *HEADING-MARGIN 3
 3 #1.  FUNCTIONAL DESCRIPTION - GENERAL
 4 & THIS IS AN EXAMPLE DOCUMENTATION SCHEMA OF THE FUNCTIONAL
 5 & DESCRIPTION FOUND IN DOD MANUAL 4120.17-M.
 6 #1.1 PURPOSE OF FUNCTIONAL DESCRIPTION
 7 #1.2 PROJECT REFERENCES
 8 #2.  SYSTEM SUMMARY
 9 #2.1 BACKGROUND/PURPOSES
10 #2.2 OBJECTIVES
11 #2.3 EXISTING METHODS AND PROCEDURES
12 #2.4 PROPOSED METHODS AND PROCEDURES
13 #2.4.1 SUMMARY OF IMPROVEMENTS
14 #2.4.2 SUMMARY OF IMPACTS
15 #2.4.2.1 EQUIPMENT IMPACTS
16 #2.4.2.2 SOFTWARE IMPACTS
17 #2.4.2.3 ORGANIZATIONAL IMPACTS
18 #2.4.2.4 OPERATIONAL IMPACTS
19 #2.4.2.5 DEVELOPMENT IMPACTS
20 #2.5 EXPECTED LIMITATIONS
21 #3.  DETAILED CHARACTERISTICS
22 #3.1 SPECIFIC PERFORMANCE REQUIREMENTS
23 #3.1.1 ACCURACY AND VALIDITY
24 #3.1.2 TIMING
25 #3.2 SYSTEMS FUNCTIONS
26 #3.3 INPUTS/OUTPUTS
27 #3.4 DATA CHARACTERISTICS
28 #3.5 FAILURE CONTINGENCIES
29 #4.  ENVIRONMENT
30 #4.1 EQUIPMENT ENVIRONMENT
31 #4.2 SUPPORT SOFTWARE ENVIRONMENT
32 #4.3 INTERFACES
33 #4.4 SECURITY
34 #5.  COST FACTORS
35 #6.  DEVELOPMENT PLAN
```

```
 1 #1.  FUNCTIONAL DESCRIPTION
 2 %SET DB=SPGEX
 3 *HEADING-SKIP 3
 4 *TOP-LINES 2
 5 *BOTTOM-LINES 2
 6 & THIS IS AN EXAMPLE DOCUMENTATION SOURCE OF THE FUNCTIONAL
 7 & DESCRIPTION STANDARDS FOUND IN DOD MANUAL 4120.17-M.
 8 & IT IS WRITTEN IN CONJUNCTION THE EXAMPLE DATA BASE FOUND
 9 & IN ISDOS WP 74.
10 *SKIP 1
11 #1.1 PURPOSE OF FUNCTIONAL DESCRIPTION
12 *SKIP 1
13 THIS FUNCTIONAL DESCRIPTION FOR THE PAYSTATEMENT EXAMPLE,
   PROJECT #1
14 IS WRITTEN TO PROVIDE:
15 A.  THE SYSTEM REQUIREMENTS TO BE SATISFIED WHICH WILL SERVE
   AS A
16 BASIS FOR MUTUAL UNDERSTANDING BETWEEN THE USER AND THE
   DEVELOPER.
17 B.  INFORMATION ON PERFORMANCE REQUIREMENTS, PRELIMINARY
   DESIGN, AND
18 USER IMPACTS, INCLUDING FIXED AND CONTINUING COSTS.
19 C.  A BASIS FOR THE DEVELOPMENT OF SYSTEM TESTS.
20 #1.2 PROJECT REFERENCES
21 *SKIP 1
22 THE PAYROLL DEPARTMENT IS THE SPONSOR AND USER OF THIS
   MANAGEMENT
23 INFORMATION SYSTEM.  THE ACTUAL OPERATION OF THIS SYSTEM
   WILL BE
24 HANDLED BY BOTH THE PAYROLL DEPARTMENT AND THE DATA
   PROCESSING
25 DEPARTMENT.
26 *SKIP 1
27 A.  A COPY OF THE PROJECT REQUEST IS IN THE APPENDIX.
28 B.  STANDARDS AND REFFERENCE DOCUMENTATION
29 1.ISDOS WORKING PAPERS 74,86,90
30 2.DOD ADS DOCUMENTATION STANDARDS MANUAL 4120.17-M
31 *HEADING-SKIP 61
32 #2.  SYSTEM SUMMARY
33 *SKIP 1
34 THIS SECTION SHALL PROVIDE A GENERAL DESCRIPTION, WRITTEN IN
35 NON-ADP TERMINOLOGY, OF THE PROPOSED ADS.
36 *HEADING-SKIP 3
37 #2.1 BACKGROUND/PURPOSES
38 *SKIP 1
39 %PCOM N=BACKGROUND-MEMO DESC NOPUNCH
40 #2.2 OBJECTIVES
41 *SKIP 1
42 %PCOM N=OBJECTIVES-MEMO DESC NOPUNCH
43 #2.3 EXISTING METHODS AND PROCEDURES
44 *SKIP 1
45 SINCE THIS IS A NEW COMPANY, THERE IS NO APPLICABLE EXISTING
46 PROCEDURE FOR DOING THE PAYROLL.
47 #2.4 PROPOSED METHODS AND PROCEDURES
```

```
48 *SKIP 1
49 %PCOM N=PROCESS-MEMO DESC NOPUNCH
50 *HOLD 43
51 %PIC N=MAIN-PROCESS NOSTRUCTURE
52 *HEADING-SKIP 2
53 #2.4.1 SUMMARY OF IMPROVEMENTS
54 *SKIP 1
55 A.  STAFFING
56 *SKIP 1
57 %PCOM N=PAYROLL-DEPT-EMPLOYEES DESC NOPUNCH
58 *SKIP 1
59 B.  TIMELINESS
60 *SKIP 1
61 %FPS NLNS NPEOF N=ONE
62 #2.4.2 SUMMARY OF IMPACTS
63 *SKIP 1
64 #2.4.2.1 EQUIPMENT IMPACTS
65 *SKIP 1
66 A LINE PRINTER CAPABLE OF PRINTING CHECKS IS NECESSARY.
67 EQUIPMENT CAPABILITIES ARE DISCUSSED IN PARAGRAPH 4.1.
68 #2.4.2.2 SOFTWARE IMPACTS
69 *SKIP 1
70 THERE ARE FIVE BASIC SOFTWARE AREAS WITHIN THE PAYROLL
   SYSTEM.
71 *SKIP 1
72 %NG S='SO=MAIN-PROCESS,1'
73 *HOLD 43
74 %PIC N=MAIN-PROCESS NODATA NOFLOW
75 #2.4.2.3 ORGANIZATIONAL IMPACTS
76 *SKIP 1
77 %FPS NLNS NPEOF N=RESPONSIBILITIES
78 *HEADING-SKIP 50
79 #2.4.2.4 OPERATIONAL IMPACTS
80 *HEADING-SKIP 2
81 *REPORT-CC OFF
82 *SKIP 1
83 A.  OPERATIONAL STRUCTURE
84 %STR PROCESS
85 *NEW-PAGE
86 B.  TIMELINESS(OPERATIONS AND DATA)
87 *SKIP 1
88 HERE IS A SUMMARY OF FREQUENCY DATA.
89 *SKIP 1
90 %NG S='INTERVAL' NOPRINT
91 %FREQ ORDER=BYTYPE INTERVAL
92 *NEW-PAGE
93 C.  INPUTS
94 %STR INPUT
95 *SKIP 2
96 D.  DATA RETENTION
97 *SKIP 1
98 %PCOM N=MASTER-FILE DESC DER NOPUNCH
99 *HOLD 43
100 %PIC N=MASTER-FILE NODATA NOFLOW
```

```
101 *HEADING-SKIP 59
102 #2.4.2.5 DEVELOPMENT IMPACTS
103 *REPORT-CC ON
104 *SKIP 1
105 %PPS NLNS NPEOF N=COMPLEXITY-LEVEL
106 *SKIP 2
107 %PPS NLNS NPEOF N=DEVELOPMENT-NEEDS
108 *HEADING-SKIP 3
109 #2.5 EXPECTED LIMITATIONS
110 *SKIP 1
111 ERRORS WILL BE NECESSARY FOR BAD DATA INPUT
112 *SKIP 1
113 %PCOM N=ERROR-LISTING DESC NOPUNCH
114 *HOLD 43
115 %PIC N=ERROR-LISTING
116 *HOLD 43
117 %PIC N=ERROR-LISTING-ENTRY
118 *HEADING-SKIP 59
119 #3.  DETAILED CHARACTERISTICS
120 *HEADING-SKIP 3
121 *SKIP 1
122 #3.1 SPECIFIC PERFORMANCE REQUIREMENTS
123 *SKIP 1
124 (SEE SECTION 2)
125 *SKIP 1
126 %PPS NLNS NPEOF N=PAYROLL-PROCESSING
127 *SKIP 2
128 DATA RECEIVED:
129 %NG S='INPUT OR SET' ORDER=BYTYPE
130 *SKIP 2
131 OUTPUTS GENERATED:
132 *SKIP 2
133 %NG S='OUTPUT'
134 #3.1.1 ACCURACY AND VALIDITY
135 *SKIP 1
136 %PCOM N=ACCURACY-MEMO DESC NOPUNCH
137 *SKIP 2
138 %PPS NLNS NPEOF N=VALIDITY-CHECK
139 *HOLD 43
140 %PIC N=HOURLY-PAYCHECK-VALIDATION
141 *NEW-PAGE
142 %PIC N=SALARIED-PAYCHECK-VALIDATION
143 #3.1.2 TIMING
144 *SKIP 1
145 THE FOLLOWING IS A DESCRIPTION OF THE FUNCTIONAL FLOW.
146 *SKIP 1
147 FOR SALARIED EMPLOYEES -
148 *SKIP 1
149 %PPS NLNS NPEOF N=SALARIED-EMP-PROCESSING-INIT
150 %PPS NLNS NPEOF N=VALIDITY-CHECK
151 %PPS NLNS NPEOF N=PASSED-ERROR-CHECKS
152 %PPS NLNS NPEOF N=SALARIED-PAYCHECK-PROD-INIT
153 *SKIP 2
154 FOR HOURLY EMPLOYEES -
```

```
155 *SKIP 1
156 %FPS NLNS NPEOF N=HOURLY-EMP-PROCESSING-INIT
157 %FPS NLNS NPEOF N=VALIDITY-CHECK
158 %FPS NLNS NPEOF N=TIME-CARD-MISSING
159 %FPS NLNS NPEOF N=PASSED-ERROR-CHECKS
160 %FPS NLNS NPEOF N=HOURLY-PAYCHECK-PROD-INIT
161 %FPS NLNS NPEOF N=NEW-EMPLOYEE-PROCESSING-INIT
162 %FPS NLNS NPEOF N=TERMINATION-PROCESSING-INIT
163 *HEADING-SKIP 59
164 #3.2 SYSTEM FUNCTIONS
165 *HEADING-SKIP 3
166 *REPORT-CC OFF
167 *SKIP 1
168 %NG S='SO=MAIN-PROCESS,1' NOPRINT
169 *SKIP 1
170 THE VARIOUS SUB-FUNCTIONS OF THE SYSTEM ARE:
171 *SKIP 1
172 %PCOM DESC PRCD NOPUNCH
173 #3.3 INPUTS/OUTPUTS
174 *SKIP 1
175 INPUTS:
176 *SKIP 1
177 %NG S='INPUT' NOPRINT
178 %FPS NLNS NPEOF
179 *SKIP 2
180 OUTPUTS:
181 *SKIP 1
182 %NG S='OUTPUT' NOPRINT
183 %FPS NLNS NPEOF
184 *SKIP 2
185 %STR OUTPUT
186 #3.4 DATA CHARACTERISTICS
187 *REPORT-CC ON
188 *SKIP 1
189 %FPS NLNS NPEOF N=PAYROLL-MASTER-INFORMATION
190 *SKIP 2
191 & THE FOLLOWING LINES ARE EXCLUSIVE TO THIS DOCUMENT
192 %PCOM N=DEPARTMENT-FILE DESC DER VOLS NOPUNCH
193 *SKIP 1
194 %PCOM N=HOURLY-EMPLOYEE-FILE DESC DER VOLM NOPUNCH
195 *HOLD 43
196 %PIC N=HOURLY-EMPLOYEE-FILE
197 *NEW-PAGE
198 %PCOM N=SALARIED-EMPLOYEE-FILE DESC DER VOLM NOPUNCH
199 *HOLD 43
200 %PIC N=SALARIED-EMPLOYEE-FILE
201 #3.5 FAILURE CHARACTERISTICS
202 *SKIP 1
203 %PCOM N=BACKUP-MEMO DESC NOPUNCH
204 *HEADING-SKIP 61
205 #4.  ENVIRONMENT
206 *HEADING-SKIP 3
207 *SKIP 1
208 #4.1 EQUIPMENT ENVIRONMENT
```

```
209 *SKIP 1
210 %PCOM N=EQUIPMENT-MEMO DESC NOPUNCH
211 #4.2 SUPPORT SOFTWARE ENVIRONMENT
212 *SKIP 2
213 %PCOM N=SOFTWARE-MEMO DESC NOPUNCH
214 #4.3 INTERFACES
215 *SKIP 1
216 %STR RWE
217 *HOLD 43
218 %PIC N=ACCOUNTING-SYSTEMS
219 *NEW-PAGE
220 %PIC N=PAYROLL-DEPARTMENT
221 #4.4 SECURITY
222 *REPORT-CC OFF
223 *SKIP 1
224 %PCOM N=SECURITY-MEMO DESC NOPUNCH
225 %NG S='SECURITY' NOPRINT
226 %FPS NLNS NPEOF
227 *HEADING-SKIP 61
228 #5.  COST FACTORS
229 %PCOM N=COSTS-MEMO DESC NOPUNCH
230 #6.  DEVELOPMENT PLAN
231 *SKIP 1
232 %PCOM N=DEVOLPMENT-MEMO DESC NOPUNCH
233 *SKIP 2
234 %FPS NLNS NPEOF N=DEVELOPMENT-TIMES
```

```
 1 *TITLE DATA REQUIREMENTS DOCUMENT
 2 *HEADING-MARGIN 3
 3 *LINES 50
 4 #1.  GENERAL DATA REQUIREMENTS
 5 & THIS IS AN EXAMPLE DOCUMENTATION SCHEM OF THE DATA
 6 & REQUIREMENTS DOCUMENT FOUND IN DOD MANUAL 4120.17-M
 7 #1.1 PURPOSE OF DATA REQUIREMENTS
 8 #1.2 PROJECT REFERENCES
 9 #1.3 MODIFICATION OF DATA REQUIREMENTS
10 #2.  DATA DESCRIPTION
11 #2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
12 #2.2 LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
13 #2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
14 #2.4 INTERNALLY GENERATED DATA
15 #2.5 SYSTEM DATA CONSTRAINTS
16 #3.  USER SUPPORT FOR DATA COLLECTION
17 #3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
18 #3.2 RECOMMEND SOURCE OF INPUT DATA
19 #3.3 DATA COLLECTION AND TRANSFER PROCEDURES
20 #3.4 DATA BASE IMPACTS
```

```
 1 #1.  GENERAL DATA REQUIREMENTS
 2 & THIS IS AN EXAMPLE DOCUMENTATION SOURCE OF THE DATA
 3 & REQUIREMENTS DOCUMENT FOUND IN DOD MANUAL 4120.17-M
 4 #1.1 PURPOSE OF DATA REQUIREMENTS
 5 *SKIP 1
 6 *SOURCE-MARGIN 1
 7 THE OBJECTIVES OF THIS DATA REQUIREMENTS DOCUMENT FOR THE
 8 PAYSTATEMENT EXAMPLE, PROJECT #1 ARE TO LIST AND DEFINE DATA
 9 ELEMENTS WHICH THE SYSTEM MUST HANDLE AND COMMUNICATE DATA
10 COLLECTION REQUIREMENTS TO THE USER.
11 #1.2 PROJECT REFERENCES
12 *SKIP 1
13 %SET DB=SPGEX
14 %PCOM N=OBJECTIVES-MEMO DESC NOPUNCH
15 *SKIP 1
16 %PCOM N=PAYROLL-DEPARTMENT DESC NOPUNCH
17 #1.3 MODIFICATION OF DATA REQUIREMENTS
18 *SKIP 1
19 NOT APPLICABLE TO THIS PROJECT.
20 *HEADING-SKIP 61
21 #2.  DATA DESCRIPTION
22 *SKIP 1
23 *SOURCE-MARGIN 5
24 *HEADING-SKIP 2
25 THE DATA DESCRIBED IN THIS SECTION SHALL BE SEPERATED INTO
   TWO
26 CATEGORIES, STATIC DATA AND DYNAMIC DATA.  STATIC DATA IS
   DEFINED
27 AS THAT DATA WHICH IS USED MAINLY FOR REFERENCE DURING
   SYSTEM OP-
28 ERATION AND IS USUALLY GENERATED OR UPDATED IN WIDELY
   SEPERATED
29 TIME FRAMES INDEPENDENT OF NORMAL SYSTEM RUNS.  DYNAMIC DATA
   IN-
30 CLUDES ALL DATA WHICH IS INTENDED TO BE UPDATED AND WHICH IS
31 INPUT TO A SYSTEM DURING A NORMAL RUN (INCLUDING "REAL TIME"
   DATA
32 SUCH AS TARGETING DATA) OR IS OUTPUT BY THE SYSTEM.  STATIC
   DATA
33 AS DESCRIBED ABOVE IS FREQUENTLY REFERRED TO AS PARAMETRIC
   DATA
34 AND DYNAMIC DATA AS NON-PARAMETRIC DATA.  BOTH, HOWEVER, ARE
   COM-
35 POSED OF DATA ELEMENTS.  THE DATA ELEMENT NAMES LISTED IN
   PARA-
36 GRAPHS 2.1,2.2, AND 2.3 SHALL BE THOSE CONTAINED IN STANDARD
37 DATA ELEMENT LIBRARIES, WHENEVER APPLICABLE.
38 #2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
39 *SKIP 1
40 %CONTENTS N=MASTER-FILE
41 *SKIP 1
42 %CONTENTS N=DEPT-FILE
43 *SKIP 1
44 %CONTENTS N=H-EMP-FILE
```

```
45 *SKIP 1
46 %CONTENTS N=S-EMP-FILE
47 #2.2 LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
48 *SKIP 1
49 *REPORT-CC OFF
50 %NG S='SO=EMP-INFO,1' NOPRINT
51 %CONTENTS
52 #2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
53 *REPORT-CC ON
54 *SKIP 1
55 %CONTENTS N=PAY-STATEMENT
56 *SKIP 1
57 %CONTENTS N=HOURLY-EMPLOYEE-REPORT
58 *SKIP 1
59 %CONTENTS N=SALARIED-EMPLOYEE-REPORT
60 *SKIP 1
61 %CONTENTS N=HIRED-EMPLOYEE-REPORT
62 *SKIP 1
63 %CONTENTS N=TERMINATED-EMPLOYEE-REPORT
64 #2.4 INTERNALLY GENERATED DATA
65 *SKIP 1
66 %CONTENTS N=ERROR-LISTING
67 #2.5 SYSTEM DATA CONSTRAINTS
68 *SKIP 1
69 %FPS N=I-O-CONSTRAINTS-MEMO
70 *HEADING-SKIP 61
71 #3.  USER SUPPORT FOR DATA COLLECTION
72 *HEADING-SKIP 2
73 %NG S='INPUT OR OUTPUT' ORDER=BYTYPE NOPRINT
74 %CNC
75 *NEW-PAGE
76 INFORMATION IN REGARDS TO DATA FLOW
77 %DP D
78 %NG S='ENTITY' NOPRINT
79 *NEW-PAGE
80 INFORMATION ON RECORDS KEPT
81 %CNC
82 *HOLD 59
83 #3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
84 *SKIP 1
85 INFORMATION NEEDED IN ORDER TO ESTABLISH THE VALUES
86 OF EACH DATA ELEMENT:
87 *SKIP 2
88 %FPS N=SALARIED-EMPLOYMENT-FORM
89 %FPS N=HOURLY-EMPLOYMENT-FORM
90 %FPS N=TAX-WITHHOLDING-CERTIFICATE
91 %FPS N=EMPLOYMENT-TERMINATION-FORM
92 *SKIP 2
93 INFORMATION TO BE COLLECTED BY THE USER:
94 *SKIP 1
95 %FPS N=TIME-CARD
96 *SKIP 2
97 OTHER SUPPLEMENTARY INFORMATION:
98 *SKIP 1
```

```
 99 A.INPUT SOURCE(S) OF THE DATA ELEMENT
100 *SKIP 1
101 %FPS N=DEPARTMENTS-AND-EMPLOYEES
102 %FPS N=EMPLOYEE
103 %FPS N=PAYROLL-DEPARTMENT
104 *SKIP 2
105 B.  RECIPIENTS
106 *SKIP 1
107 %FPS N=ACCOUNTING-SYSTEMS
108 *SKIP 2
109 D.  CRITICAL VALUES
110 %FPS N=REMAINING-FUNDS
111 *SKIP 2
112 E.  OUTPUT FORM/DEVICE
113 *SKIP 1
114 %PCOM N=H-EMP-REPORT DESC NOPUNCH
115 *SKIP 1
116 %PCOM N=S-EMP-REPORT DESC NOPUNCH
117 *SKIP 2
118 F.  FREQUENCY OF UPDATE
119 %FREQ
120 #3.2 RECOMMENDED SOURCE OF INPUT DATA
121 *SKIP 1
122 THIS TOPIC HAS BEEN DISCUSSED ABOVE(SECTION 3.1.A)
123 #3.3 DATA COLLECTION AND TRANSFER PROCEDURES
124 *SKIP 1
125 THIS TOPIC HAS BEEN DISCUSSED ABOVE(SECTION 3.1.F)
126 #3.4 DATA BASE IMPACTS
127 *SKIP 1
128 %PCOM N=DEPARTMENT-FILE DESC DER VOLS NOPUNCH
129 %PCOM N=HOURLY-EMPLOYEE-FILE DESC DER VOLM NOPUNCH
130 %PCOM N=SALARIED-EMPLOYEE-FILE DESC DER VOLM NOPUNCH
```

Appendix G

DOCUMENT EXAMPLES

AD-A057 130 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS ANALYZER (URA) USER'S MANUAL H6180/MULTICS/VE--ETC(U)
MAR 77 D TEICHROEW, A HERSHEY, S SPEWAK F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-128 NL

6 OF 6
AD A057130



END
DATE FILMED
9-79
DDC

This appendix contains examples of the Functional Description Document written using the Automated Documentation System. The Analyzer data base used is the Pay System example described in Part IV of this manual. Due to space limitations, listings of the Analyzer data base are not included.

The following list of Analyzer commands have been used to produce these examples:

Command Name	Parameters Used
CONTENTS	FILE NAME
DATA PROCESS	DATA FILE
FPS	NAME FILE PRINT
FREQUENCY	
NAME-GEN	ENTITY INPUT OUTPUT PRINT NOPRINT SUBLEVEL SUBPART OF SECURITY
PICTURE	DATA NODATA FILE NAME FLOW NOFLOW STRUCTURE NOSTRUCTURE
PUNCH-COMMENT-ENTRY	DER DESC PROC VOLM VOLS FILE NAME NOPUNCH PRINT
STRUCTURE	INPUT PROCESS RWE

OUTPUT

Although these are the only commands used, all Analyzer commands may be used with the Automated Documentation Generator (with the exception of the STOP command).

Appendix H

EXECUTING AUTOMATIC DOCUMENTATION SYSTEM

Using Multics, the Automated Documentation System can run using the following command:

```
exec-com >ml>CARA>spg                    schema-segment source-segment
```

This command will do all syntax checking needed and generate the document. Output for the document will go to the file spg.output. The system assumes the Analyzer data base is in the file uradb.dbf unless the SET command is used to change the default (i.e., SET DB=file).

Output from this section looks as follows:

```
Line  TEXT  CMNDS SECTION

   2                &
   3                & A SMALL TEST EXAMPLE
   4                &
   5   0      0     =1.  SCOPE
   6   0      2     =1.1 IDENTIFICATION
   7   0      1     =1.2 FUNCTIONAL SUMMARY
   8   *MISSING*    =1.3
   9                =1.4
*** INVALID SECTION IDENTIFIER
  10                &
  11   0      0     =2.  APPLICABLE DOCUMENTS
  12   13     2     =2.1 GOVERNMENT DOCUMENTS
  13   5      1     =2.2 NON-GOVERNMENT DOCUMENTS


SUMMARY
   8 SECTIONS
   1 OF WHICH ARE MISSING FROM THE SOURCE FILE.
  18 TOTAL TEST LINES, AND
   6 TOTAL COMMAND LINES.
```

GLOSSARY

analyst	Name used synonymously for "problem definer." One who aids to develop the problem statement or logical system design.
Analyzer	Synonym for "URA." Is the software package that processes problem stated in the Language.
comment entry	The text associated with a comment entry statement. DESCRIPTION, PROCEDURE and VOLATILITY are examples of statements which are specified by comment entries.
comment entry statement	Any URL statement in which the contents of the statement are defined by the problem definer (as is narrative description). The DESCRIPTION and PROCEDURE statements are examples of this.
control commands	Those URA commands which allow the URA user to pass certain control information to the Analyzer. They are particular to an individual operating system.
conversational mode	Interactive use of the computer system through a terminal device. Used synonymously with on-line terminal, or interactive mode.
data base	The data base referred to throughout this paper is the user's data base which is populated by URA from the user inputted URL statements.
data object	Any URA name type that represents some form of data. SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS are all data objects described by URL.
fdname	Any legal file or device name. Allowable names are dependent on the operating system being used.
filename	Any legal temporary or permanent file name that is to be specified by the user.
input file	Any temporary or permanent file which contains data to be used by URA commands via INPUT or FILE parameters.
logical description	Synonym to "problem statement." Set of requirements for a new system.
logical system design	The process of specifying a problem statement for any particular system.

modifier commands	Commands which modify the contents of a URA data base in the manner specified by the user.
Multics Command	A command to the Multics system; an element of the Multics command language.
Multics Command Language	The set of commands, subcommands and operands recognized by the Multics Operating System.
name type	Any of the many types of names allowed by URL (i.e., PROCESS, SET, GROUP, etc.). See "The User Requirements Language, Language Reference Manual"[1], Appendix E, for a list of all possible name types.
physical system design	The process of specifying a physical system (consisting of software, machinery, etc.) given a particular problem statement.
physical system designer	Person responsible for deriving a physical system design from the problem statement generated from the logical system design process.
problem definer	Used synonymously with "analyst." That person who develops the requirements stated by the users into a format understandable by others and in sufficient detail to be usable by the physical system designer. The product of his work is the problem statement.
problem statement	A set of requirements specified by users of a proposed system and expressed by the problem definer into a format acceptable by the organization.
prompt	A system function that requests the terminal user to supply operands necessary to continue processing.
PUNCH file	A file which contains data (usually URL user-names) in a format that can be used as input to one or more URA commands.
report commands	Those commands which retrieve data from a URA data base and output it in some meaningful format.
segment	A named collection of data which is accessible

---

[1] Part II of URL User's Manual.

	by the system. The data set usually resides on an auxiliary storage device.
term	Identifier to designate that the terminal is to be used as a source of input or area for output.
undefined name	A name of a URL object that has been entered into the URA data base, but has no name type associated with it.
URA	The User Requirements Analyzer. A software package which stores information in the URA data base by interpretting URL statements given as input, and retrieves information from the data base in the form of reports.
URA command	Any of the commands that can be used to operate URA. See "User Requirements Analyzer Command Descriptions"[1] for complete descriptions about each command available in URA.
URA data base	File where URL information is stored (in a coded format) which can then be accessed by URA user.
URL	The User Requirements Language. The collection of all URL statements allowed for use by URA. See "The User Requirements Language, Language Reference Manual"[2] for complete description.
URL statement	A statement specified by "The User Requirements Language, Language Reference Manual".[2] Each statement may define a URL object, define a comment entry, or define a relationship among two or more URL objects.
user-name	Any legal URL name specified by problem definer. Also called a "user defined name".

---

[1] Part IV.
[2] Part II of URL User's Manual.

Appendix I

ASCII CHARACTER SET FOR URA

The attached list gives for each ASCII character a code of 1 to 4 classifying the characters into the following categories:

Code 1: Nonprinting operating System and transaission control characters to be treated as punctuation, but will always be illegal.

Code 2: Puncutation, delimiters, etc. which are not allowed in names.

Code 3: Characters allowed at any position in a name.

Code 4: Characters allowed at any position in a name after the first.

There are three versions of this categorization:

1. A one page summary.
2. Sorted by Octal representation.
3. Sorted by code, then by Octal representation.

CODE 1: All others

CODE 2: "&'()*,:;=?[ ]|¬

CODE 3: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
!#$@ {}

CODE 4: 0123456789
+-./<>_

CODE	OCTAL	CHAR	NAME
1	000	nul	null or time fill char
1	001	soh	start of heading
1	002	stx	start of text
1	003	etx	end of text (EOM)
1	004	eot	end of transmission (EOT)
1	005	enq	enquiry (WRU)
1	006	ack	acknowledge (RU)
1	007	bel	bell
1	010	bs	backspace
1	011	ht	horizontal tabulation (TAB)
1	012	lf	line feed (LINE FEED)
1	013	vt	vertical tabulation (VT)
1	014	ff	form feed (FORM)
1	015	cr	carriage return (RETURN)
1	016	so	shift out
1	017	si	shift in
1	020	dle	data link escape
1	021	dc1	device control 1 (X-ON)
1	022	dc2	device control 2 (TAPE)
1	023	dc3	device control 3 (X-OFF)
1	024	dc4	device control 4 (TAPE)
1	025	nak	negative acknowledge
1	026	syn	synchronous idle
1	027	etb	end of transmission blocks
1	030	can	cancel
1	031	em	end of medium
1	032	ss	special sequence
1	033	esc	escape
1	034	fs	file separator
1	035	gs	group separator
1	036	rs	record separator
1	037	us	unit separator

CODE	OCTAL	CHAR	NAME
2	040	sp	space (SPACE BAR)
3	041	!	exclamation point
2	042	"	quotation mark
3	043	#	number sign
3	044	$	currency symbol
3	045	%	percent
2	046	&	ampersand
2	047	'	apostrophe
2	050	(	opening parenthesis
2	051	)	closing parenthesis
2	052	*	asterisk
4	053	+	plus
2	054	,	comma
4	055	-	hyphen or minus
4	056	.	period
4	057	/	slant
4	060	0	zero
4	061	1	one
4	062	2	two
4	063	3	three
4	064	4	four
4	065	5	five
4	066	6	six
4	067	7	seven
4	070	8	eight
4	071	9	nine
2	072	:	colon
2	073	;	semicolon
4	074	<	less than
2	075	=	equal
4	076	>	greater than
2	077	?	question mark

CODE	OCTAL	CHAR	NAME
3	100	@	commercial at
3	101	A	
3	102	B	
3	103	C	
3	104	D	
3	105	E	
3	106	F	
3	107	G	
3	110	H	
3	111	I	
3	112	J	
3	113	K	
3	114	L	
3	115	M	
3	116	N	
3	117	O	
3	120	P	
3	121	Q	
3	122	R	
3	123	S	
3	124	T	
3	125	U	
3	126	V	
3	127	W	
3	130	X	
3	131	Y	
3	132	Z	
2	133	[	opening bracket
3	134	\	reverse slant
2	135	]	closing bracket
3	136	^	circumflex
4	137	_	underline

CODE	OCTAL	CHAR	NAME
3	140	`	grave accent
3	141	a	
3	142	b	
3	143	c	
3	144	d	
3	145	e	
3	146	f	
3	147	g	
3	150	h	
3	151	i	
3	152	j	
3	153	k	
3	154	l	
3	155	m	
3	156	n	
3	157	o	
3	160	p	
3	161	q	
3	162	r	
3	163	s	
3	164	t	
3	165	u	
3	166	v	
3	167	w	
3	170	x	
3	171	y	
3	172	z	
3	173	{	opening brace
2	174	|	vertical line
3	175	}	closing brace
2	176	~	tilde
1	177	del	delete (RUBOUT)

CODE	OCTAL	CHAR	NAME
1	000	nul	null or time fill char
1	001	soh	start of heading
1	002	stx	start of text
1	003	etx	end of text (EOM)
1	004	eot	end of transmission (EOT)
1	005	enq	enquiry (WRU)
1	006	ack	acknowledge (RU)
1	007	bel	bell
1	010	bs	backspace
1	011	ht	horizontal tabulation (TAB)
1	012	lf	line feed (LINE FEED)
1	013	vt	vertical tabulation (VT)
1	014	ff	form feed (FORM)
1	015	cr	carriage return (RETURN)
1	016	so	shift out
1	017	si	shift in
1	020	dle	data link escape
1	021	dc1	device control 1 (X-ON)
1	022	dc2	device control 2 (TAPE)
1	023	dc3	device control 3 (X-OFF)
1	024	dc4	device control 4 (TAPE)
1	025	nak	negative acknowledge
1	026	syn	synchronous idle
1	027	etb	end of transmission blocks
1	030	can	cancel
1	031	em	end of medium
1	032	ss	special sequence
1	033	esc	escape
1	034	fs	file separator
1	035	gs	group separator
1	036	rs	record separator
1	037	us	unit separator
1	177	del	delete (RUBOUT)

CODE	OCTAL	CHAR	NAME
2	040	sp	space (SPACE BAR)
2	042	"	quotation mark
2	046	&	ampersand
2	047	'	apostrophe
2	050	(	opening parenthesis
2	051	)	closing parenthesis
2	052	*	asterisk
2	054	,	comma
2	072	:	colon
2	073	;	semicolon
2	075	=	equal
2	077	?	question mark
2	133	[	opening bracket
2	135	]	closing bracket
2	174	|	vertical line
2	176	~	tilde

CODE	OCTAL	CHAR	NAME
3	041	!	exclamation point
3	043	#	number sign
3	044	$	currency symbol
3	045	%	percent
3	100	@	commercial at
3	101	A	
3	102	B	
3	103	C	
3	104	D	
3	105	E	
3	106	F	
3	107	G	
3	110	H	
3	111	I	
3	112	J	
3	113	K	
3	114	L	
3	115	M	
3	116	N	
3	117	O	
3	120	P	
3	121	Q	
3	122	R	
3	123	S	
3	124	T	
3	125	U	
3	126	V	
3	127	W	
3	130	X	
3	131	Y	
3	132	Z	

CODE	OCTAL	CHAR	NAME
3	134		reverse slant
3	136		circumflex
3	140		grave accent
3	141	a	
3	142	b	
3	143	c	
3	144	d	
3	145	e	
3	146	f	
3	147	g	
3	150	h	
3	151	i	
3	152	j	
3	153	k	
3	154	l	
3	155	m	
3	156	n	
3	157	o	
3	160	p	
3	161	q	
3	162	r	
3	163	s	
3	164	t	
3	165	u	
3	166	v	
3	167	w	
3	170	x	
3	171	y	
3	172	z	
3	173	{	opening brace
3	175	}	closing brace

CODE	OCTAL	CHAR	NAME
4	053	+	plus
4	055	-	hyphen or minus
4	056	.	period
4	057	/	slant
4	060	0	zero
4	061	1	one
4	062	2	two
4	063	3	three
4	064	4	four
4	065	5	five
4	066	6	six
4	067	7	seven
4	070	8	eight
4	071	9	nine
4	074	<	less than
4	076	>	greater than
4	137	_	underline

Printed by
United States Air Force
Hanscom AFB, Mass. 01731